**FTC CSN 3.0**

CSN Quick Search Design with Cloudera Solr

**Lockheed Martin Corporation**

**June 16, 2016**





Contents

[**1** **Introduction** 3](#_Toc448685714)

[**2** **Solr Cloud Dev Architecture** 3](#_Toc448685715)

[**3** **Environmental Setup** 5](#_Toc448685716)

[**3.1** **Runtime SOLR Configuration** 5](#_Toc448685717)

[3.1.1 Types of Field 5](#_Toc448685718)

[3.1.2 NGRAM Tokenizer 6](#_Toc448685719)

[3.1.3 Synonyms 6](#_Toc448685720)

[3.1.4 Copy Fields 7](#_Toc448685721)

[3.1.5 Validation 7](#_Toc448685722)

[**3.2** **Create SOLR Collection** 7](#_Toc448685723)

[**3.3** **Create SOLR Core** 8](#_Toc448685724)

[**3.4** **Update SOLR Configuration** 8](#_Toc448685725)

[**4** **Quick Search Configuration** 8](#_Toc448685726)

[**4.1** **Schema.XML** 8](#_Toc448685727)

[**4.2** **Quick Search Queries** 68](#_Toc448685728)

[**5** **Real Time Indexing using Flume & Morphlines** 69](#_Toc448685729)

[**5.1** **Configure Flume sql-source** 70](#_Toc448685730)

[**5.2** **Morphline Configuration** 71](#_Toc448685731)

[**6** **Initial Indexing** 82](#_Toc448685732)

[**7** **Incremental Indexing** 82](#_Toc448685733)

[**8** **Performance Improvement** 83](#_Toc448685734)

[**Appendix 1) Existing FAST Configurations** 84](#_Toc448685735)

[Quick Search Implementation 84](#_Toc448685736)

[Composite Fields 84](#_Toc448685737)

[Composite Field – ***Content*** 84](#_Toc448685738)

[Composite Field – ContentWithBaseName 86](#_Toc448685739)

[Composite Field – ***CFAlertsIDT*** 87](#_Toc448685740)

[Composite Field – ***CFAlertsCIS*** 88](#_Toc448685741)

[Rank Profile 89](#_Toc448685742)

[Default Rank Profile 89](#_Toc448685743)

[Navigators 90](#_Toc448685744)

[Indexing 96](#_Toc448685745)

[CIS 96](#_Toc448685746)

[IDT Pipeline 99](#_Toc448685747)

[DNC Pipeline 102](#_Toc448685748)

[Processors 104](#_Toc448685749)

# **Introduction**

The purpose of this document is to highlight the design elements to implement the Sentinel Quick Search feature using Cloudera Search.

# **Solr Cloud Dev Architecture**

SOLR is running on Hadoop cluster. SOLR indices are distributed among different HDFS data nodes. Cloudera 5.5 being used as a Hadoop distributor. Three SOLR nodes FTCDEVSPARK01 – 03 are being used. Zoo Keeper Service manages the SOLR instances and port on which they are running to track leader and replicas. Collections can be configured to be spread over one or more shards. Each collection returns complete result set. In case where collection is spread over two or more active shards, each shard returns a part of the result set. This improves the query response time.



Exhibit 1: Cloudera Solr Cloud Dev Architecture

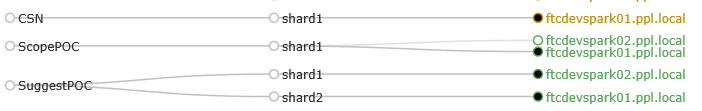


Exhibit 2: Representation of Physical Deployments

# **Environmental Setup**

## **Runtime SOLR Configuration**

In order to start using SOLR for indexing the data, we must configure a collection holding the index. A configuration for a collection requires a solrconfig.xml file, a schema.xml and any helper files may be referenced from the xml files. The solrconfig.xml file contains all of the SOLR settings for a given collection, and the schema.xml file specifies the schema that SOLR uses when indexing documents. Configuration files for a collection are managed as part of the instance directory. An empty instance directory can be created using following command

*$ solrctl instancedir --generate $HOME/solr\_configs*

SOLRCTL command is the preferred option with CLOUDERA SOLR while managing collections.

Then we can edit and customize it by directly editing the solrconfig.xml and schema.xml files that have been created in $HOME/solr\_configs/conf. Once we are satisfied with the configuration, we will make it available for SOLR to use by issuing the following command, which uploads the content of the entire instance directory to ZooKeeper:

*$ solrctl instancedir --create collection1 $HOME/solr\_configs*

Highlights of the schema.xml. Original schema.xml can be referred in Appendix 1.

### Types of Field

For existing CSN search capabilities we will be needing following types of fields -

* <**fieldType name="string" class="solr.StrField" sortMissingLast="true"**/>
* <**fieldType name="boolean" class="solr.BoolField" sortMissingLast="true"**/>
* <**fieldType name="int" class="solr.TrieIntField" precisionStep="0" positionIncrementGap="0"**/>
* <**fieldType name="float" class="solr.TrieFloatField" precisionStep="0" positionIncrementGap="0"**/>
* <**fieldType name="ngram" class="solr.TextField" sortMissingLast="true"**>

<**analyzer**>

<**tokenizer class="solr.NGramTokenizerFactory" minGramSize="6" maxGramSize="14"**/>

</**analyzer**>

</**fieldType**>

<fieldType name="text\_syn" class="solr.TextField" sortMissingLast="true">

<analyzer type="query">

<tokenizer class="solr.WhitespaceTokenizerFactory"/>

<filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true" expand="false"/>

<filter class="solr.LowerCaseFilterFactory"/>

</analyzer>

<analyzer>

<tokenizer class="solr.WhitespaceTokenizerFactory" />

<filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true" expand="false"/>

<filter class="solr.LowerCaseFilterFactory"/>

</analyzer>

</fieldType>

### NGRAM Tokenizer

NGram Tokenizers are available OOTB where we can define minimum NGram Size to Max NGram Size. Based on experience an optimum range of 6 to 14 is used for NGram Size. An example of NGram tokenizer is illustrated below –

“Bank of America” is tokenized to have following tokens (not all ) –

"Bank o", "Bank of", "Bank of ", "Bank of A", "Bank of Am", "Bank of Ame", "Bank of Amer", "Bank of Ameri"

### Synonyms

Synonyms are captured as part of synonym configuration file (dictionary). The dictionary can support one directional conversion and expansion. For CSN Name Normalization single directional conversion is desired. The format is simple and as below

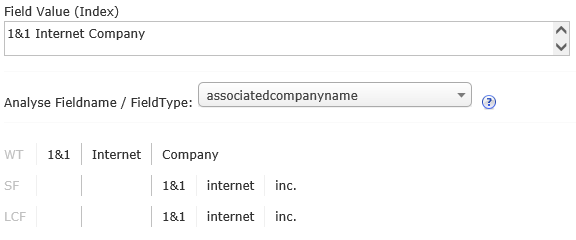
Variation => Base Form

The dictionary can be named anything of choice. There can be multiple synonyms dictionary per instance. Synonym field types are defined by specifying analyzers for query and indexing time. Optionally synonym behavior during query time and indexing time can be different. Analyzer for synonyms accepts the synonym dictionary name as an attribute.

With following entry in the synonym dictionary

1&1 Internet Company => 1&1 Internet Inc.

For “1&1 Internet Company” as the company name the entry is converted as below to its base form



### Copy Fields

As for fields like company names we need to support both synonyms match as well as NGram, same cannot be accommodated within a single file. We therefore create additional fields for NGram and copy the value of original field to this field using Copy Fields.

### Validation

We can use the solrctl tool to verify that our instance directory uploaded successfully and is available to ZooKeeper. We can use the solrctl to list the contents of an instance directory as follows:

*$ solrctl instancedir --list*

the --list command should return collection1.

## **Create SOLR Collection**

By default, the SOLR server comes up with no collections. We create our collection using the instancedir that we provided to SOLR in previous steps by using the same collection name.

*$ solrctl collection --create collection1*

We should be able to check that the collection is active. For example, for the server ftcdevspark01.ppl.local, we should be able to navigate to http://ftcdevspark01.ppl.local:8983/solr/collection1/select?q=\*%3A\*&wt=json&indent=true and verify that the collection is active. Similarly, we should also be able to observe the topology of our SolrCloud using a URL similar to: [http://](http://myhost.example.com:8983/solr/#/~cloud)ftcdevspark01.ppl.local[:8983/solr/#/~cloud](http://myhost.example.com:8983/solr/#/~cloud)

## **Create SOLR Core**

We can execute following command on a given SolrCloud node to create a core.

*$ solrctl core --create CSN*

## **Update SOLR Configuration**

We can update or modify an existing SolrCloud's copy of an instance directory based on the files present in a local filesystem. This can be thought of first using --delete collection1 followed by --create collection1 path.

*$ solrctl instancedir --reload collection1*

Then we also have to reload core using following command.

*$ solrctl core --reload CSN*

# **Quick Search Configuration**

## **Schema.XML**

<?xml version="1.0" encoding="UTF-8"?>

<schema name="CSN" version="1.5">

<fields>

<!-- Fields for CISMain, IDTMain, DNCMain collections -->

<!-- CIS, IDT, DNC, Alerts -->

<!--<field name="documentid" index="no" result="no"/>-->

<field name="documentid" type="string" required="true"/>

<!-- CIS, IDT, DNC, Alerts -->

<!--<field name="recordid" type="int32" index="no" result="no"/>-->

<field name="recordid" type="int" indexed="false" stored="false"/>

<!-- CIS, IDT, DNC -->

<!--<field name="referencenumber" result="dynamic" fallback-ref="referencenumber" fullsort="yes"/>-->

<field name="referencenumber" type="string"/>

<!--<field name="intreferencenumber" type="int32" result="dynamic" fallback-ref="intreferencenumber" fullsort="yes"/>-->

<field name="intreferencenumber" type="int"/>

<!-- CIS -->

<!--<field name="contacttype" boundary-match="yes" result="dynamic" fallback-ref="contacttype"/>-->

<field name="contacttype" type="text\_ws"/>

<!-- CIS, IDT -->

<!--<field name="internetrelatedflag" result="dynamic"/>-->

<field name="internetrelatedflag" type="boolean"/>

<!-- CIS, IDT -->

<!--<field name="language" boundary-match="yes"/>-->

<field name="language" type="text\_ws"/>

<!-- CIS -->

<!--<field name="source" result="dynamic" fallback-ref="source" boundary-match="yes"/>-->

<field name="source" type="text\_ws"/>

<!-- CIS -->

<!--<field name="dncflag" index="no"/>-->

<field name="dncflag" type="boolean" indexed="false"/>

<!-- CIS, IDT, DNC, Alerts -->

<!--<field name="comments" result="dynamic" fallback-ref="comments"/>-->

<field name="comments" type="string"/>

<!-- IDT Part II -->

<!--<field name="comments2" result="dynamic" fallback-ref="comments2"/>-->

<field name="comments2" type="string"/>

<!-- CIS, IDT, DNC, Alerts -->

<!--<field name="commentsteaser" index="no" result="dynamic" fallback-ref="commentsteaser"/>-->

<field name="commentsteaser" type="string" indexed="false"/>

<!-- CIS, IDT, DNC, Alerts -->

<!--<field name="redactedcomments" index="no" result="static" fallback-ref="redactedcomments"/>-->

<field name="redactedcomments" type="string" indexed="false"/>

<!-- CIS, IDT, DNC, Alerts -->

<!--<field name="redactedcommentsteaser" index="no" result="dynamic" fallback-ref="redactedcommentsteaser"/>-->

<field name="redactedcommentsteaser" type="string" indexed="false"/>

<!-- CIS, IDT, DNC, Alerts -->

<!--<field name="createddate" type="datetime" result="dynamic" fallback-ref="createddate" sort="yes"/>-->

<field name="createddate" type="date"/>

<!-- CIS, IDT, DNC -->

<!--<field name="externalrefnumber" result="dynamic" fallback-ref="externalrefnumber"/>-->

<field name="externalrefnumber" type="string"/>

<!-- CIS -->

<!--<field name="enteredby" index="no"/>-->

<field name="enteredby" type="string" indexed="false"/>

<!-- CIS -->

<!--<field name="updatedby" index="no"/>-->

<field name="updatedby" type="string" indexed="false"/>

<!-- CIS, IDT -->

<!--<field name="updateddate" type="datetime"/>-->

<field name="updateddate" type="date"/>

<!-- CIS, IDT -->

<!--<field name="quicksearchflag" type="string"/>-->

<field name="quicksearchflag" type="string"/>

<!-- IDT -->

<!--<field name="occurreddate" index="no" type="datetime"/>-->

<field name="occurreddate" type="date" indexed="false"/>

<!-- IDT -->

<!--<field name="noticeddate" index="no" type="datetime"/>-->

<field name="noticeddate" type="date" indexed="false"/>

<!-- IDT Part II-->

<!--<field name="numberaccountsaffected" result="dynamic" fallback-ref="numberaccountsaffected" type="float"/>-->

<field name="numberaccountsaffected" type="float"/>

<!-- IDT -->

<!--<field name="theftviainternet" index="yes"/>-->

<field name="theftviainternet" type="string"/>

<!-- IDT -->

<!--<field name="thefttype" separator="|" result="dynamic" fallback-ref="thefttype"/>-->

<field name="thefttype" type="string" multiValued="true"/>

<!-- IDT Part II-->

<!--<field name="thefttypenav" separator="|" result="dynamic" fallback-ref="thefttype"/>-->

<field name="thefttypenav" type="string" multiValued="true"/>

<!-- IDT -->

<!--<field name="theftsubtype" separator="|" result="dynamic" fallback-ref="theftsubtype"/>-->

<field name="theftsubtype" type="string" multiValued="true"/>

<!-- IDT -->

<!--<field name="harmsuffered" separator="|" result="dynamic" fallback-ref="harmsuffered"/>-->

<field name="harmsuffered" type="string" multiValued="true"/>

<!-- CIS, IDT, DNC -->

<!--<field name="organization" result="dynamic" fallback-ref="organization" boundary-match="yes" fullsort="yes"/>-->

<field name="organization" type="text\_ws"/>

<!-- CIS, IDT -->

<!--<field name="moneypaid" result="no"/>-->

<field name="moneypaid" type="string" stored="false"/>

<!-- CIS -->

<!--<field name="amountrequested" type="float" result="dynamic" fallback-ref="amountrequested"/>-->

<field name="amountrequested" type="float"/>

<!-- CIS -->

<!--<field name="amountpaid" type="float" result="dynamic" fallback-ref="amountpaid" sort="yes"/>-->

<field name="amountpaid" type="float"/>

<!-- CIS -->

<!--<field name="paymentmethod" boundary-match="yes" result="dynamic" fallback-ref="paymentmethod"/>-->

<field name="paymentmethod" type="text\_ws"/>

<!-- CIS -->

<!--<field name="agencycontact" result="dynamic" fallback-ref="agencycontact" boundary-match="yes"/>-->

<field name="agencycontact" type="text\_ws"/>

<!-- CIS -->

<!--<field name="initialcontact" boundary-match="yes" result="dynamic" fallback-ref="initialcontact"/>-->

<field name="initialcontact" type="text\_ws"/>

<!-- CIS -->

<!--<field name="initialresponse" boundary-match="yes" result="dynamic" fallback-ref="initialresponse"/>-->

<field name="initialresponse" type="text\_ws"/>

<!-- CIS, DNC -->

<!--<field name="complaintdate" type="datetime" index="no"/>-->

<field name="complaintdate" type="date" indexed="false"/>

<!-- DNC -->

<!--<field name="complaintchannel" index="no"/>-->

<field name="complaintchannel" type="string" indexed="false"/>

<!-- DNC -->

<!--<field name="dncexistingbusinessrel" result="dynamic" fallback-ref="dncexistingbusinessrel"/>-->

<field name="dncexistingbusinessrel" type="string"/>

<!-- DNC -->

<!--<field name="dncprerecmessage" result="dynamic" fallback-ref="dncprerecmessage"/>-->

<field name="dncprerecmessage" type="string"/>

<!-- DNC -->

<!--<field name="dncrequeststopcalling" result="dynamic" fallback-ref="dncrequeststopcalling"/>-->

<field name="dncrequeststopcalling" type="string"/>

<!-- CIS, DNC -->

<!--<field name="transactiondate" type="datetime" result="dynamic" fallback-ref="transactiondate"/>-->

<field name="transactiondate" type="date"/>

<!-- CIS -->

<!--<field name="topic" separator="|" result="dynamic" fallback-ref="topic"/>-->

<field name="topic" type="string" multiValued="true"/>

<!-- CIS -->

<!--<field name="topicnav" separator="|"/>-->

<field name="topicnav" type="string" multiValued="true"/>

<!-- CIS, IDT, DNC -->

<!--<field name="prodservicedesc" separator="|" result="dynamic" fallback-ref="prodservicedesc"/>-->

<field name="prodservicedesc" type="string" multiValued="true"/>

<!-- CIS, IDT, DNC -->

<!--<field name="prodservicedescnav" separator="|"/>-->

<field name="prodservicedescnav" type="string" multiValued="true"/>

<!-- CIS -->

<!--<field name="statute" separator="|" result="dynamic" fallback-ref="statute"/>-->

<field name="statute" type="string" multiValued="true"/>

<!-- CIS -->

<!--<field name="statutenav" separator="|"/>-->

<field name="statutenav" type="string" multiValued="true"/>

<!-- CIS -->

<!--<field name="violation" separator="|" result="dynamic" fallback-ref="violation"/>-->

<field name="violation" type="string" multiValued="true"/>

<!-- CIS -->

<!--<field name="violationnav" separator="|"/>-->

<field name="violationnav" type="string" multiValued="true"/>

<!-- CIS -->

<!--<field name="representativefname" index="no" separator="|"/>-->

<field name="representativefname" type="string" indexed="false" multiValued="true"/>

<!-- CIS -->

<!--<field name="representativelname" separator="|" result="dynamic" fallback-ref="representativelname"/>-->

<field name="representativelname" type="string" multiValued="true"/>

<!-- CIS -->

<!--<field name="representativetitle" index="no" separator="|"/>-->

<field name="representativetitle" type="string" indexed="false" multiValued="true"/>

<!-- CIS -->

<!--<field name="complainingcompany" separator="|" result="dynamic" fallback-ref="complainingcompany"/>-->

<field name="complainingcompany" type="string" multiValued="true"/>

<!-- CIS, IDT Part II -->

<!--<field name="complaintagainstcreditbureau" result="dynamic" fallback-ref="complaintagainstcreditbureau" separator="|"/>-->

<field name="complaintagainstcreditbureau" type="string"/>

<!-- CIS -->

<!--<field name="complaintagainstcreditbureau45days"/>-->

<field name="complaintagainstcreditbureau45days" type="string"/>

<!-- IDT Part II -->

<!--<field name="lawviolationcodes" result="dynamic" fallback-ref="lawviolationcodes" separator="|"/>-->

<field name="lawviolationcodes" type="string" multiValued="true"/>

<!-- start nested document -->

<!-- CIS, IDT, DNC, Alerts -->

<!--<field name="consumerlname" separator="|" result="dynamic" fallback-ref="consumerlname"/>-->

<field name="consumerlname" type="string"/>

<!-- CIS, IDT, DNC, Alerts -->

<!--<field name="consumerfname" separator="|" result="dynamic" fallback-ref="consumerfname"/>-->

<field name="consumerfname" type="string"/>

<!-- IDT Part II -->

<!--<field name="consumermname" separator="|" result="dynamic" fallback-ref="consumermname"/>-->

<field name="consumermname" type="string"/>

<!-- CIS, IDT, DNC, Alerts -->

<!--<field name="consumeraddr1" separator="|" result="dynamic" fallback-ref="consumeraddr1"/>-->

<field name="consumeraddr1" type="string"/>

<!-- IDT Part II -->

<!--<field name="consumeraddr2" separator="|" result="dynamic" fallback-ref="consumeraddr2"/>-->

<field name="consumeraddr2" type="string"/>

<!-- CIS, IDT, DNC, Alerts -->

<!--<field name="consumercity" separator="|" result="dynamic" fallback-ref="consumercity"/>-->

<field name="consumercity" type="string"/>

<!-- IDT Part II -->

<!--<field name="consumercitynav" separator="|"/>-->

<field name="consumercitynav" type="string"/>

<!-- IDT Part II -->

<!--<field name="consumerstate" separator="|" result="dynamic" fallback-ref="consumerstate"/>-->

<field name="consumerstate" type="string"/>

<!-- IDT Part II -->

<!--<field name="consumerstatenav" separator="|"/>-->

<field name="consumerstatenav" type="string"/>

<!--<field name="consumerzip" separator="|" result="dynamic" fallback-ref="consumerzip"/>-->

<field name="consumerzip" type="string"/>

<!-- CIS, IDT, DNC, Alerts -->

<!--<field name="consumerzipnav" separator="|"/>-->

<field name="consumerzipnav" type="string"/>

<!-- CIS, IDT, Alerts -->

<!--<field name="consumercountry" separator="|" result="dynamic" fallback-ref="consumercountry"/>-->

<field name="consumercountry" type="string"/>

<!-- IDT Part II -->

<!--<field name="consumerlivedheresince" index="no" separator="|"/>-->

<field name="consumerlivedheresince" type="string" indexed="false"/>

<!-- CIS, IDT, Alerts -->

<!--<field name="consumerworkphone" separator="|" result="dynamic" fallback-ref="consumerworkphone"/>-->

<field name="consumerworkphone" type="string"/>

<!-- CIS, IDT, Alerts -->

<!--<field name="consumerworkphoneext" separator="|" result="dynamic" fallback-ref="consumerworkphoneext"/>-->

<field name="consumerworkphoneext" type="string"/>

<!-- CIS, DNC -->

<!--<field name="consumerfaxnumber" separator="|" index="no"/>-->

<field name="consumerfaxnumber" type="string"/>

<!-- IDT -->

<!--<field name="consumerhomenumber" separator="|" result="dynamic" fallback-ref="consumerhomenumber"/>-->

<field name="consumerhomenumber" type="string"/>

<!-- IDT -->

<!--<field name="consumercellnumber" separator="|" result="dynamic" fallback-ref="consumercellnumber"/>-->

<field name="consumercellnumber" type="string"/>

<!-- CIS, IDT, Alerts -->

<!--<field name="consumeremail" separator="|" result="dynamic" fallback-ref="consumeremail"/>-->

<field name="consumeremail" type="string"/>

<!-- CIS -->

<!--<field name="consumeragerange" separator="|"/>-->

<field name="consumeragerange" type="string"/>

<!-- CIS -->

<!--<field name="consumeragerangenav" separator="|"/>-->

<field name="consumeragerangenav" type="string"/>

<!-- CIS, Alerts -->

<!--<field name="consumercountrycode" separator="|" result="no"/>-->

<field name="consumercountrycode" type="string" stored="false"/>

<!-- CIS, DNC, Alerts -->

<!--<field name="consumerareacode" separator="|"/>-->

<field name="consumerareacode" type="string"/>

<!-- IDT, Alerts -->

<!--<field name="consumerssn" separator="|" result="no"/>-->

<field name="consumerssn" type="string" stored="false"/>

<!-- IDT Part II -->

<!--<field name="consumeridnumber" separator="|" result="no"/>-->

<field name="consumeridnumber" type="string"/>

<!-- IDT Part II -->

<!--<field name="consumeridissuedstate" separator="|" result="dynamic" fallback-ref="consumeridissuedstate"/>-->

<field name="consumeridissuedstate" type="string"/>

<!-- IDT Part II -->

<!--<field name="soldierstatus" separator="|" result="dynamic" fallback-ref="soldierstatus"/>-->

<field name="soldierstatus" type="string"/>

<!-- IDT Part II -->

<!--<field name="soldierstation" separator="|" result="dynamic" fallback-ref="soldierstation"/>-->

<field name="soldierstation" type="string"/>

<!-- IDT Part II -->

<!--<field name="soldierpaygrade" separator="|" result="dynamic" fallback-ref="soldierpaygrade"/>-->

<field name="soldierpaygrade" type="string"/>

<!-- IDT -->

<!--<field name="consumerdateofbirth" separator="|" result="dynamic" fallback-ref="consumerdateofbirth"/>-->

<field name="consumerdateofbirth" type="string"/>

<!-- DNC -->

<!--<field name="isphoneinregistry" separator="|" result="dynamic" fallback-ref="isphoneinregistry"/>-->

<field name="isphoneinregistry" type="string"/>

<!-- end nested document -->

<!-- IDT Part II -->

<!--<field name="showcatffields" index="no"/>-->

<field name="showcatffields" type="string" indexed="false"/>

<!-- IDT Part II -->

<!--<field name="catffname" index="no"/>-->

<field name="catffname" type="string" indexed="false"/>

<!-- IDT Part II -->

<!--<field name="catfmname" index="no"/>-->

<field name="catfmname" type="string" indexed="false"/>

<!-- IDT Part II -->

<!--<field name="catflname" index="no"/>-->

<field name="catflname" type="string" indexed="false"/>

<!-- IDT Part II -->

<!--<field name="catfaddress1" index="no"/>-->

<field name="catfaddress1" type="string" indexed="false"/>

<!-- IDT Part II -->

<!--<field name="catfaddress2" index="no"/>-->

<field name="catfaddress2" type="string" indexed="false"/>

<!-- IDT Part II -->

<!--<field name="catfcity" index="no"/>-->

<field name="catfcity" type="string" indexed="false"/>

<!-- IDT Part II -->

<!--<field name="catfcountry" index="no"/>-->

<field name="catfcountry" type="string" indexed="false"/>

<!-- IDT Part II -->

<!--<field name="catfstate" index="no"/>-->

<field name="catfstate" type="string" indexed="false"/>

<!-- IDT Part II -->

<!--<field name="catfzip" index="no"/>-->

<field name="catfzip" type="string" indexed="false"/>

<!-- IDT Part II -->

<!--<field name="catflivedherefrom" index="no"/>-->

<field name="catflivedherefrom" type="string" indexed="false"/>

<!-- IDT Part II -->

<!--<field name="catflivedhereto" index="no"/>-->

<field name="catflivedhereto" type="string" indexed="false"/>

<!-- IDT Part II -->

<!--<field name="catfphone" index="no"/>-->

<field name="catfphone" type="string" indexed="false"/>

<!-- IDT Part II -->

<!--<field name="showreportedbyfields" index="no"/>-->

<field name="showreportedbyfields" type="string" indexed="false"/>

<!-- IDT Part II -->

<!--<field name="reportedbybusinessname" index="no"/>-->

<field name="reportedbybusinessname" type="string" indexed="false"/>

<!-- IDT Part II -->

<!--<field name="reportedbymname" index="no"/>-->

<field name="reportedbymname" type="string" indexed="false"/>

<!-- start nested document -->

<!-- IDT Part II-->

<!--<field name="creditbureauname" separator="|" result="dynamic" fallback-ref="creditbureauname"/>-->

<field name="creditbureauname" type="string"/>

<!-- IDT Part II-->

<!--<field name="creditbureaunotified" separator="|" result="dynamic" fallback-ref="creditbureaunotified"/>-->

<field name="creditbureaunotified" type="string"/>

<!-- IDT -->

<!--<field name="associatedcompanycontactname" separator="|" index="no"/>-->

<field name="associatedcompanycontactname" type="string" indexed="false"/>

<!-- end nested document -->

<!-- IDT -->

<!--<field name="reportedbylname" index="no"/>-->

<field name="reportedbylname" type="string" indexed="false"/>

<!-- IDT -->

<!--<field name="reportedbyfname" index="no"/>-->

<field name="reportedbyfname" type="string" indexed="false"/>

<!-- IDT -->

<!--<field name="reportedbyaddr1" index="no"/>-->

<field name="reportedbyaddr1" type="string" indexed="false"/>

<!-- IDT -->

<!--<field name="reportedbyaddr2" index="no"/>-->

<field name="reportedbyaddr2" type="string" indexed="false"/>

<!-- IDT -->

<!--<field name="reportedbycity" index="no"/>-->

<field name="reportedbycity" type="string" indexed="false"/>

<!-- IDT -->

<!--<field name="reportedbystate" index="no"/>-->

<field name="reportedbystate" type="string" indexed="false"/>

<!-- IDT -->

<!--<field name="reportedbyzip" index="no"/>-->

<field name="reportedbyzip" type="string" indexed="false"/>

<!-- IDT -->

<!--<field name="reportedbycountry" index="no"/>-->

<field name="reportedbycountry" type="string" indexed="false"/>

<!-- IDT -->

<!--<field name="reportedbyworkphone" index="no"/>-->

<field name="reportedbyworkphone" type="string" indexed="false"/>

<!-- IDT -->

<!--<field name="reportedbyworkphoneext" index="no"/>-->

<field name="reportedbyworkphoneext" type="string" indexed="false"/>

<!-- IDT -->

<!--<field name="reportedbyhomenumber" index="no"/>-->

<field name="reportedbyhomenumber" type="string" indexed="false"/>

<!-- IDT -->

<!--<field name="reportedbyemail" index="no"/>-->

<field name="reportedbyemail" type="string" indexed="false"/>

<!-- IDT Part II-->

<!--<field name="reportedbyfax" index="no"/>-->

<field name="reportedbyfax" type="string" indexed="false"/>

<!-- IDT Part II-->

<!--<field name="reportedbycell" index="no"/>-->

<field name="reportedbycell" type="string" indexed="false"/>

<!-- start nested document -->

<!-- IDT, Alerts -->

<!--<field name="suspectfname" separator="|" result="dynamic" fallback-ref="suspectfname"/>-->

<field name="suspectfname" type="string"/>

<!-- IDT, Alerts -->

<!--<field name="suspectlname" separator="|" result="dynamic" fallback-ref="suspectlname"/>-->

<field name="suspectlname" type="string"/>

<!-- IDT Part II, Alerts -->

<!--<field name="suspectmname" separator="|" result="dynamic" fallback-ref="suspectmname"/>-->

<field name="suspectmname" type="string"/>

<!-- IDT Part II-->

<!--<field name="suspectvictimrelationship" result="dynamic" fallback-ref="suspectvictimrelationship" separator="|"/>-->

<field name="suspectvictimrelationship" type="string"/>

<!-- IDT Part II-->

<!--<field name="suspectobtainmethod" result="dynamic" fallback-ref="suspectobtainmethod" separator="|"/>-->

<field name="suspectobtainmethod" type="string"/>

<!-- IDT -->

<!--<field name="suspectaddnlinfo" index="no" separator="|"/>-->

<field name="suspectaddnlinfo" type="string" indexed="false"/>

<!-- end nested document -->

<!-- start nested document -->

<!-- CIS, IDT (suspectcompanyname), DNC, , Alerts -->

<!--<field name="primarycompanyname" separator="|" result="dynamic" fallback-ref="primarycompanyname"/>-->

<field name="primarycompanyname" type="string"/>

<!-- CIS, IDT (suspectcompanyaddr1), Alerts -->

<!--<field name="primarycompanyaddr1" separator="|" result="dynamic" fallback-ref="primarycompanyaddr1" wildcard="full"/>-->

<field name="primarycompanyaddr1" type="string"/>

<!-- CIS, IDT (suspectcompanyaddr2), -->

<!--<field name="primarycompanyaddr2" separator="|" result="dynamic" fallback-ref="primarycompanyaddr2"/>-->

<field name="primarycompanyaddr2" type="string"/>

<!-- CIS, IDT (suspectcompanycity), Alerts-->

<!--<field name="primarycompanycity" separator="|" result="dynamic" fallback-ref="primarycompanycity"/>-->

<field name="primarycompanycity" type="string"/>

<!-- CIS, IDT (suspectcompanystate), DNC, , Alerts -->

<!--<field name="primarycompanystate" separator="|" result="dynamic" fallback-ref="primarycompanystate"/>-->

<field name="primarycompanystate" type="string"/>

<!-- CIS, IDT (suspectcompanyzip), Alerts -->

<!--<field name="primarycompanyzip" separator="|" result="dynamic" fallback-ref="primarycompanyzip"/>-->

<field name="primarycompanyzip" type="string"/>

<!-- IDT Part II -->

<!--<field name="primarycompanyzipnav" separator="|"/>-->

<field name="primarycompanyzipnav" type="string"/>

<!-- CIS, IDT (suspectcompanycountry), Alerts -->

<!--<field name="primarycompanycountry" separator="|" result="dynamic" fallback-ref="primarycompanycountry"/>-->

<field name="primarycompanycountry" type="string"/>

<!-- CIS, IDT (suspectcompanyemail), Alerts -->

<!--<field name="primarycompanyemail" separator="|" result="dynamic" fallback-ref="primarycompanyemail"/>-->

<field name="primarycompanyemail" type="string"/>

<!-- CIS, IDT (suspectcompanyurl), Alerts -->

<!--<field name="primarycompanyurl" separator="|" result="dynamic" fallback-ref="primarycompanyurl"/>-->

<field name="primarycompanyurl" type="string"/>

<!-- CIS, IDT, DNC, Alerts -->

<!--<field name="primarycompanycountrycode" separator="|" result="dynamic" fallback-ref="primarycompanycountrycode"/>-->

<field name="primarycompanycountrycode" type="string"/>

<!-- CIS, IDT, DNC, Alerts -->

<!--<field name="primarycompanyareacode" separator="|" result="dynamic" fallback-ref="primarycompanyareacode"/>-->

<field name="primarycompanyareacode" type="string"/>

<!-- CIS, IDT, DNC, Alerts -->

<!--<field name="primarycompanyphonenumber" separator="|" result="dynamic" fallback-ref="primarycompanyphonenumber"/>-->

<field name="primarycompanyphonenumber" type="string"/>

<!-- IDT Part II -->

<!--<field name="primarycompanyphonenumberqs" separator="|" result="dynamic" fallback-ref="primarycompanyphonenumberqs"/>-->

<field name="primarycompanyphonenumberqs" type="string"/>

<!-- CIS, IDT, Alerts -->

<!--<field name="primarycompanyphoneext" separator="|" index="no"/>-->

<field name="primarycompanyphoneext" type="string" indexed="false"/>

<!-- end nested document -->

<!-- start nested document -->

<!-- CIS, IDT -->

<!--<field name="associatedcompanyname" separator="|" result="dynamic" fallback-ref="associatedcompanyname"/>-->

<field name="associatedcompanyname" type="string"/>

<!-- IDT -->

<!--<field name="associatedcompanynamenav" separator="|"/>-->

<field name="associatedcompanynamenav" type="string"/>

<!-- IDT -->

<!--<field name="associatedcompanybasename" separator="|" result="dynamic" fallback-ref="associatedcompanybasename"/>-->

<field name="associatedcompanybasename" type="string"/>

<!-- IDT -->

<!--<field name="associatedcompanybasenamenav" separator="|"/>-->

<field name="associatedcompanybasenamenav" type="string"/>

<!-- CIS, IDT -->

<!--<field name="associatedcompanytype" separator="|" result="dynamic" fallback-ref="associatedcompanytype"/>-->

<field name="associatedcompanytype" type="string"/>

<!-- IDT -->

<!--<field name="associatedcompanytypenav" separator="|"/>-->

<field name="associatedcompanytypenav" type="string"/>

<!-- CIS, IDT, Alerts -->

<!--<field name="associatedcompanyaddr1" separator="|" result="dynamic" fallback-ref="associatedcompanyaddr1"/>-->

<field name="associatedcompanyaddr1" type="string"/>

<!-- CIS, IDT -->

<!--<field name="associatedcompanyaddr2" separator="|" result="dynamic" fallback-ref="associatedcompanyaddr2"/>-->

<field name="associatedcompanyaddr2" type="string"/>

<!-- CIS, IDT, Alerts -->

<!--<field name="associatedcompanycity" separator="|" result="dynamic" fallback-ref="associatedcompanycity"/>-->

<field name="associatedcompanycity" type="string"/>

<!-- IDT -->

<!--<field name="associatedcompanycitynav" separator="|"/>-->

<field name="associatedcompanycitynav" type="string"/>

<!-- CIS, IDT, Alerts -->

<!--<field name="associatedcompanystate" separator="|" result="dynamic" fallback-ref="associatedcompanystate"/>-->

<field name="associatedcompanystate" type="string"/>

<!-- IDT -->

<!--<field name="associatedcompanystatenav" separator="|"/>-->

<field name="associatedcompanystatenav" type="string"/>

<!-- CIS, IDT, Alerts -->

<!--<field name="associatedcompanyzip" separator="|" result="dynamic" fallback-ref="associatedcompanyzip"/>-->

<field name="associatedcompanyzip" type="string"/>

<!-- CIS, IDT, Alerts -->

<!--<field name="associatedcompanycountry" separator="|" result="dynamic" fallback-ref="associatedcompanycountry"/>-->

<field name="associatedcompanycountry" type="string"/>

<!-- IDT -->

<!--<field name="associatedcompanycountrynav" separator="|"/>-->

<field name="associatedcompanycountrynav" type="string"/>

<!-- CIS, Alerts -->

<!--<field name="associatedcompanyemail" separator="|" result="dynamic" fallback-ref="associatedcompanyemail"/>-->

<field name="associatedcompanyemail" type="string"/>

<!-- CIS, Alerts -->

<!--<field name="associatedcompanyurl" separator="|" result="dynamic" fallback-ref="associatedcompanyurl"/>-->

<field name="associatedcompanyurl" type="string"/>

<!-- CIS, Alerts -->

<!--<field name="associatedcompanycountrycode" separator="|" result="dynamic" fallback-ref="associatedcompanycountrycode"/>-->

<field name="associatedcompanycountrycode" type="string"/>

<!-- CIS, Alerts -->

<!--<field name="associatedcompanyareacode" separator="|" result="dynamic" fallback-ref="associatedcompanyareacode"/>-->

<field name="associatedcompanyareacode" type="string"/>

<!-- CIS, Alerts -->

<!--<field name="associatedcompanyphonenumber" separator="|" result="dynamic" fallback-ref="associatedcompanyphonenumber"/>-->

<field name="associatedcompanyphonenumber" type="string"/>

<!-- CIS, Alerts -->

<!--<field name="associatedcompanyphoneext" separator="|" result="dynamic" fallback-ref="associatedcompanyphoneext"/>-->

<field name="associatedcompanyphoneext" type="string"/>

<!-- IDT Part II -->

<!--<field name="associatedemail" separator="|" result="dynamic" fallback-ref="associatedemail"/>-->

<field name="associatedemail" type="string"/>

<!-- IDT Part II -->

<!--<field name="associatedurl" separator="|" result="dynamic" fallback-ref="associatedurl"/>-->

<field name="associatedurl" type="string"/>

<!-- end nested document -->

<!-- start nested document -->

<!-- IDT -->

<!--<field name="institutiontype" separator="|" result="dynamic" fallback-ref="institutiontype"/>-->

<field name="institutiontype" type="string"/>

<!--<field name="institutiontypenav" separator="|"/>-->

<field name="institutiontypenav" type="string"/>

<!-- IDT -->

<!--<field name="institutionnotified" index="no" separator="|"/>-->

<field name="institutionnotified" type="string" indexed="false"/>

<!-- IDT -->

<!--<field name="institutionwrittennotification" index="no" separator="|"/>-->

<field name="institutionwrittennotification" type="string" indexed="false"/>

<!-- IDT Part II-->

<!--<field name="institutionfraudalertset" result="dynamic" fallback-ref="institutionfraudalertset" separator="|"/>-->

<field name="institutionfraudalertset" type="string"/>

<!-- IDT Part II-->

<!--<field name="institutioncreditreportordered" result="dynamic" fallback-ref="institutioncreditreportordered" separator="|"/>-->

<field name="institutioncreditreportordered" type="string"/>

<!-- CIS, IDT -->

<!--<field name="problemwithcompany" separator="|"/>-->

<field name="problemwithcompany" type="string"/>

<!-- IDT -->

<!--<field name="institutionpsc" separator="|" result="dynamic" fallback-ref="institutionpsc"/>-->

<field name="institutionpsc" type="string"/>

<!-- IDT -->

<!--<field name="institutionpscnav" separator="|"/>-->

<field name="institutionpscnav" type="string"/>

<!-- IDT -->

<!--<field name="institutionsc" separator="|" result="dynamic" fallback-ref="institutionsc"/>-->

<field name="institutionsc" type="string"/>

<!-- IDT -->

<!--<field name="institutionscnav" separator="|"/>-->

<field name="institutionscnav" type="string"/>

<!-- IDT -->

<!--<field name="institutionvc" separator="|" result="dynamic" fallback-ref="institutionvc"/>-->

<field name="institutionvc" type="string"/>

<!-- IDT -->

<!--<field name="institutionvcnav" separator="|"/>-->

<field name="institutionvcnav" type="string"/>

<!-- IDT -->

<!--<field name="idtverifiedcomplaint" separator="|"/>-->

<field name="idtverifiedcomplaint" type="string"/>

<!-- end nested document -->

<!-- start nested document -->

<!-- IDT -->

<!--<field name="lawenfdeptname" separator="|" result="dynamic" fallback-ref="lawenfdeptname"/>-->

<field name="lawenfdeptname" type="string"/>

<!-- IDT -->

<!--<field name="lawenfcontactperson" index="no" separator="|"/>-->

<field name="lawenfcontactperson" type="string" indexed="false"/>

<!-- IDT -->

<!--<field name="lawenfdeptstate" separator="|" result="dynamic" fallback-ref="lawenfdeptstate"/>-->

<field name="lawenfdeptstate" type="string"/>

<!-- IDT Part II-->

<!--<field name="lawenforcementphone" result="dynamic" fallback-ref="lawenforcementphone" separator="|"/>-->

<field name="lawenforcementphone" type="string"/>

<!-- IDT Part II-->

<!--<field name="notifiedlocallawenforcement" result="dynamic" fallback-ref="notifiedlocallawenforcement" separator="|"/>-->

<field name="notifiedlocallawenforcement" type="string"/>

<!-- IDT Part II -->

<!--<field name="lawenfreporttaken" result="dynamic" fallback-ref="lawenfreporttaken" separator="|"/>-->

<field name="lawenfreporttaken" type="string"/>

<!-- IDT -->

<!--<field name="lawenfreportnumber" separator="|" result="dynamic" fallback-ref="lawenfreportnumber"/>-->

<field name="lawenfreportnumber" type="string"/>

<!-- IDT Part II-->

<!--<field name="lawenfreporttakendate" separator="|" index="no"/>-->

<field name="lawenfreporttakendate" type="string" indexed="false"/>

<!-- end nested document -->

<!-- start nested document -->

<!-- CIS, IDT -->

<!--<field name="subjectname" separator="|" result="dynamic" fallback-ref="subjectname"/>-->

<field name="subjectname" type="string"/>

<!-- CIS, IDT -->

<!--<field name="subjectcity" separator="|" result="dynamic" fallback-ref="subjectcity"/>-->

<field name="subjectcity" type="string"/>

<!-- CIS, IDT -->

<!--<field name="subjectstate" separator="|" result="dynamic" fallback-ref="subjectstate"/>-->

<field name="subjectstate" type="string"/>

<!-- CIS, IDT -->

<!--<field name="subjectcountry" separator="|" result="dynamic" fallback-ref="subjectcountry"/>-->

<field name="subjectcountry" type="string"/>

<!-- CIS, IDT created by a document processor-->

<!--<field name="subjectphone" separator="|" result="dynamic" fallback-ref="subjectphone"/>-->

<field name="subjectphone" type="string"/>

<!-- CIS, IDT created by a document processor-->

<!--<field name="subjectphonenav" separator="|" result="dynamic" fallback-ref="subjectphonenav"/>-->

<field name="subjectphonenav" type="string"/>

<!-- CIS, IDT -->

<!--<field name="subjectemail" separator="|" result="dynamic" fallback-ref="subjectemail"/>-->

<field name="subjectemail" type="string"/>

<!-- CIS, IDT -->

<!--<field name="subjectemaildomain" separator="|" result="dynamic" fallback-ref="subjectemaildomain"/>-->

<field name="subjectemaildomain" type="string"/>

<!-- CIS, IDT -->

<!--<field name="subjecturl" separator="|" result="dynamic" fallback-ref="subjecturl"/>-->

<field name="subjecturl" type="string"/>

<!-- CIS, IDT -->

<!--<field name="subjectlocation" separator="|" result="dynamic" fallback-ref="subjectlocation"/>-->

<field name="subjectlocation" type="string"/>

<!-- CIS, IDT -->

<!--<field name="subjectaddress" separator="|" result="dynamic" fallback-ref="subjectaddress" wildcard="full"/>-->

<field name="subjectaddress" type="string"/>

<!-- CIS, IDT -->

<!--<field name="subjectaddressoriginal" separator="|" result="dynamic" fallback-ref="subjectaddressoriginal" wildcard="full"/>-->

<field name="subjectaddressoriginal" type="string"/>

<!-- CIS-->

<!--<field name="subjectid" separator="|" result="no"/>-->

<field name="subjectid" type="string"/>

<!-- CIS-->

<!--<field name="subjectidtype" separator="|" result="dynamic" fallback-ref="subjectidtype"/>-->

<field name="subjectidtype" type="string"/>

<!-- CIS-->

<!--<field name="subjectidissuerstate" separator="|" result="dynamic" fallback-ref="subjectidissuerstate"/>-->

<field name="subjectidissuerstate" type="string"/>

<!-- CIS-->

<!--<field name="subjectidissuercountry" separator="|" result="dynamic" fallback-ref="subjectidissuercountry"/>-->

<field name="subjectidissuercountry" type="string"/>

<!-- CIS-->

<!--<field name="complaintresolutionflag" separator="|" result="dynamic" fallback-ref="complaintresolutionflag"/>-->

<field name="complaintresolutionflag" type="boolean"/>

<!-- CIS-->

<!--<field name="complaintresolutiontext" separator="|" result="dynamic" fallback-ref="complaintresolutiontext"/>-->

<field name="complaintresolutiontext" type="string"/>

<!-- CIS-->

<!--<field name="subjectidissuertype" separator="|" result="dynamic" fallback-ref="subjectidissuertype"/>-->

<field name="subjectidissuertype" type="string"/>

<!-- end nested document -->

<!-- start nested document -->

<!-- Fields specific for Incidents search -->

<!-- IDT Part II-->

<!--<field name="routingnumber" separator="|" result="dynamic" fallback-ref="routingnumber"/>-->

<field name="routingnumber" type="string"/>

<!-- IDT Part II-->

<!--<field name="accountnumber" separator="|" result="dynamic" fallback-ref="accountnumber"/>-->

<field name="accountnumber" type="string"/>

<!-- IDT Part II-->

<!--<field name="checknumbers" separator="|" result="dynamic" fallback-ref="checknumbers"/>-->

<field name="checknumbers" type="string"/>

<!-- IDT Part II-->

<!--<field name="incidentoccurreddate" separator="|" result="dynamic" fallback-ref="incidentoccurreddate" type="datetime"/>-->

<field name="incidentoccurreddate" type="date"/>

<!-- end nested document -->

<!-- start nested document -->

<!-- Fields specific for Theft Details -->

<!-- IDT Part II-->

<!--<field name="authorizetouseyourid" separator="|" index="no"/>-->

<field name="authorizetouseyourid" type="string" indexed="false"/>

<!-- IDT Part II-->

<!--<field name="receiveanybenefits" separator="|" index="no"/>-->

<field name="receiveanybenefits" type="string" indexed="false"/>

<!-- IDT Part II-->

<!--<field name="willingtopresscharges" separator="|" index="no"/>-->

<field name="willingtopresscharges" type="string" indexed="false"/>

<!-- IDT Part II-->

<!--<field name="inaccuratepersonalinfo1" separator="|" index="no"/>-->

<field name="inaccuratepersonalinfo1" type="string" indexed="false"/>

<!-- IDT Part II-->

<!--<field name="inaccuratepersonalinfo2" separator="|" index="no"/>-->

<field name="inaccuratepersonalinfo2" type="string" indexed="false"/>

<!-- IDT Part II-->

<!--<field name="inaccuratepersonalinfo3" separator="|" index="no"/>-->

<field name="inaccuratepersonalinfo3" type="string" indexed="false"/>

<!-- IDT Part II-->

<!--<field name="creditinquirycompany1" separator="|" index="no"/>-->

<field name="creditinquirycompany1" type="string" indexed="false"/>

<!-- IDT Part II-->

<!--<field name="creditinquirycompany2" separator="|" index="no"/>-->

<field name="creditinquirycompany2" type="string" indexed="false"/>

<!-- IDT Part II-->

<!--<field name="creditinquirycompany3" separator="|" index="no"/>-->

<field name="creditinquirycompany3" type="string" indexed="false"/>

<!-- end nested document -->

<!-- Non Moneytory Harm -->

<!-- IDT Part II-->

<!--<field name="harmcomments" separator="|" index="no"/>-->

<field name="harmcomments" type="string" indexed="false" multiValued="true"/>

<!-- start nested document -->

<!-- Fields required for Alerts search -->

<!-- Alerts -->

<!--<field name="alertcreatorfirstname" index="no"/>-->

<field name="alertcreatorfirstname" type="string" indexed="false"/>

<!-- Alerts -->

<!--<field name="alertcreatorlastname" index="no"/>-->

<field name="alertcreatorlastname" type="string" indexed="false"/>

<!-- Alerts -->

<!--<field name="alertoriginalownerfirstname" result="dynamic" fallback-ref="alertoriginalownerfirstname"/>-->

<field name="alertoriginalownerfirstname" type="string"/>

<!-- Alerts -->

<!--<field name="alertoriginalownerlastname" result="dynamic" fallback-ref="alertoriginalownerlastname"/>-->

<field name="alertoriginalownerlastname" type="string"/>

<!-- Alerts -->

<!--<field name="alertoriginalowneremail" index="no"/>-->

<field name="alertoriginalowneremail" type="string" indexed="false"/>

<!-- Alerts -->

<!--<field name="alertdelegatedownerfirstname" separator="|" result="dynamic" fallback-ref="alertdelegatedownerfirstname"/>-->

<field name="alertdelegatedownerfirstname" type="string"/>

<!-- Alerts -->

<!--<field name="alertdelegatedownerlastname" separator="|" result="dynamic" fallback-ref="alertdelegatedownerlastname"/>-->

<field name="alertdelegatedownerlastname" type="string"/>

<!-- Alerts -->

<!--<field name="alertdelegatedowneremail" index="no" separator="|"/>-->

<field name="alertdelegatedowneremail" type="string" indexed="false"/>

<!-- Alerts -->

<!--<field name="alertpocfirstname" result="dynamic" fallback-ref="alertpocfirstname"/>-->

<field name="alertpocfirstname" type="string"/>

<!-- Alerts -->

<!--<field name="alertpoclastname" result="dynamic" fallback-ref="alertpoclastname"/>-->

<field name="alertpoclastname" type="string"/>

<!-- Alerts -->

<!--<field name="alertcontactmethod" index="no"/>-->

<field name="alertcontactmethod" type="string" indexed="false"/>

<!-- Alerts -->

<!--<field name="alertcontactvalue" index="no"/>-->

<field name="alertcontactvalue" type="string" indexed="false"/>

<!-- Alerts -->

<!--<field name="alertname" result="dynamic" fallback-ref="alertname"/>-->

<field name="alertname" type="string"/>

<!-- Alerts -->

<!--<field name="alertexpirationdate" type="datetime" index="no"/>-->

<field name="alertexpirationdate" type="string" indexed="false"/>

<!-- Alerts -->

<!--<field name="alertmaxduraton" type="datetime" index="no"/>-->

<field name="alertmaxduraton" type="string" indexed="false"/>

<!-- Alerts -->

<!--<field name="alerttypeofbusines" separator="|" result="dynamic" fallback-ref="alerttypeofbusines"/>-->

<field name="alerttypeofbusines" type="string" multiValued="true"/>

<!-- Alerts -->

<!--<field name="alertcreatorid" index="no"/>-->

<field name="alertcreatorid" type="string" indexed="false"/>

<!-- Alerts -->

<!--<field name="alertoriginalownerid" index="no"/>-->

<field name="alertoriginalownerid" type="string" indexed="false"/>

<!-- Alerts -->

<!--<field name="alertdelegatedownerid" index="no"/>-->

<field name="alertdelegatedownerid" type="string" indexed="false"/>

<!-- Alerts -->

<!--<field name="alertpocid" index="no"/>-->

<field name="alertpocid" type="string" indexed="false"/>

<!-- Alerts -->

<!--<field name="recordtype" result="dynamic" fallback-ref="recordtype" fullsort="yes"/>-->

<field name="recordtype" type="string"/>

<!-- Alerts -->

<!--<field name="alertid"/>-->

<field name="creditinquirycompany3" type="string"/>

<!-- end nested document -->

<!-- Fields required for Mail Scanning-->

<!--<field name="mailimagefilename" result="dynamic" fallback-ref="mailimagefilename"/>-->

<field name="mailimagefilename" type="string"/>

<!--<field name="hasmailscannedimage" result="dynamic" fallback-ref="hasmailscannedimage"/>-->

<field name="hasmailscannedimage" type="string"/>

<!--<field name="hasaudiofiles" result="dynamic" fallback-ref="hasaudiofiles"/>-->

<field name="hasaudiofiles" type="string"/>

<!-- Fields required for FAQ search -->

<!-- FAQ -->

<!--<field name="faqid"/>-->

<field name="faqid" type="string"/>

<!-- FAQ -->

<!--<field name="faqquestion" lemmatize="yes" result="dynamic" fallback-ref="faqquestion"/>-->

<field name="faqquestion" type="text\_general"/>

<!-- FAQ -->

<!--<field name="faqanswer" lemmatize="yes" result="dynamic" fallback-ref="faqanswer"/>-->

<field name="faqanswer" type="text\_general"/>

<!-- FAQ -->

<!--<field name="faqtopic" lemmatize="yes" result="dynamic" fallback-ref="faqtopic"/>-->

<field name="faqtopic" type="text\_general"/>

<!-- FAQ -->

<!--<field name="faqteaser" index="no" result="dynamic" fallback-ref="faqteaser"/>-->

<field name="faqteaser" type="string" indexed="false"/>

<!-- FAQ -->

<!--<field name="faqtopiccode"/>-->

<field name="faqtopiccode" type="string"/>

<!-- Fields required for Security Access Manager SAM (secure search). -->

<!-- All Collections -->

<!--<field name="docacl"/>-->

<field name="docacl" type="string"/>

<!-- All Collections -->

<!--<field name="docaclsystemid"/>-->

<field name="docaclsystemid" type="string"/>

<!-- All Collections -->

<!--<field name="docaclisencoded"/>-->

<field name="docaclisencoded" type="string"/>

<!-- start nested document -->

<!-- Fields required for accommodating subject name variations -->

<!--<field name="subjectnamevariations" separator="|" result="dynamic" fallback-ref="subjectnamevariations"/>-->

<field name="subjectnamevariations" type="string"/>

<!--<field name="subjectbasename" separator="|" result="dynamic" fallback-ref="subjectbasename"/>-->

<field name="subjectbasename" type="string"/>

<!--<field name="subjectnamestripped" separator="|" result="dynamic" fallback-ref="subjectnamestripped"/>-->

<field name="subjectnamestripped" type="string"/>

<!--<field name="subjectbasenamestripped" separator="|" result="dynamic" fallback-ref="subjectbasenamestripped"/>-->

<field name="subjectbasenamestripped" type="string"/>

<!--<field name="subjectbasenamestrippedphonetic" separator="|" result="dynamic" fallback-ref="subjectbasenamestrippedphonetic"/>-->

<field name="subjectbasenamestrippedphonetic" type="string"/>

<!-- Field to indicate FCRA Flag -->

<!--<field name="cradisputeresponded" index="no" separator="|"/>-->

<field name="cradisputeresponded" type="boolean" indexed="false"/>

<!-- Field to indicate FCRA Flag -->

<!--<field name="cradisputeresolved" index="no" separator="|"/>-->

<field name="cradisputeresolved" type="string" indexed="false"/>

<!--<field name="matchedbasename" separator="|" result="dynamic" fallback-ref="matchedbasename"/>-->

<field name="matchedbasename" type="string"/>

<!--<field name="matchedassociatedbasename" separator="|" result="dynamic" fallback-ref="matchedassociatedbasename"/>-->

<field name="matchedassociatedbasename" type="string"/>

<!-- end nested document -->

<!-- Standard fields -->

<!-- All Collections -->

<!--<field name="processingtime" type="datetime" fullsort="yes"/>-->

<field name="processingtime" type="date"/>

<!-- All Collections -->

<!--<field name="docvector" index="no"/>-->

<field name="docvector" type="string" indexed="false"/>

<!-- Standard fields, generic placeholders. -->

<!--<field name="generic1" result="dynamic" fallback-ref="generic1"/>-->

<field name="generic1" type="string"/>

<!--<field name="generic2" result="dynamic" fallback-ref="generic2"/>-->

<field name="generic2" type="string"/>

<!--<field name="generic3" result="dynamic" fallback-ref="generic3"/>-->

<field name="generic3" type="string"/>

<!--<field name="generic4" result="no"/>-->

<field name="generic4" type="string" indexed="false"/>

<!--<field name="igeneric1" type="int32" fullsort="yes" result="dynamic" fallback-ref="igeneric1"/>-->

<field name="igeneric1" type="int"/>

<!--<field name="igeneric2" type="int32" fullsort="yes" result="dynamic" fallback-ref="igeneric2"/>-->

<field name="igeneric2" type="int"/>

<!--<field name="dtgeneric1" type="datetime" fullsort="yes" result="dynamic" fallback-ref="dtgeneric1"/>-->

<field name="dtgeneric1" type="date"/>

<!--<field name="dtgeneric2" type="datetime" fullsort="yes" result="dynamic" fallback-ref="dtgeneric2"/>-->

<field name="dtgeneric2" type="date"/>

<!--<field name="phonetic1" result="dynamic" fallback-ref="phonetic1"/>-->

<field name="phonetic1" type="string"/>

<!--<field name="phonetic2" result="dynamic" fallback-ref="phonetic2"/>-->

<field name="phonetic2" type="string"/>

<!--<field name="phonetic3" result="dynamic" fallback-ref="phonetic3"/>-->

<field name="phonetic3" type="string"/>

<!--<field name="phonetic4" result="dynamic" fallback-ref="phonetic4"/>-->

<field name="phonetic4" type="string"/>

<!--<field name="phonetic5" result="dynamic" fallback-ref="phonetic5"/>-->

<field name="phonetic5" type="string"/>

<!--<field name="phonetic6" result="dynamic" fallback-ref="phonetic6"/>-->

<field name="phonetic6" type="string"/>

<!--<field name="phonetic7" result="dynamic" fallback-ref="phonetic7"/>-->

<field name="phonetic7" type="string"/>

<!--<field name="phonetic8" result="dynamic" fallback-ref="phonetic8"/>-->

<field name="phonetic8" type="string"/>

<!--<field name="phonetic9" result="dynamic" fallback-ref="phonetic9"/>-->

<field name="phonetic9" type="string"/>

<!--<field name="phonetic10" result="dynamic" fallback-ref="phonetic10"/>-->

<field name="phonetic10" type="string"/>

<!--<field name="phonetic11" result="dynamic" fallback-ref="phonetic11"/>-->

<field name="phonetic11" type="string"/>

<!--<field name="phonetic12" result="dynamic" fallback-ref="phonetic12"/>-->

<field name="phonetic12" type="string"/>

<!-- Entity extraction placeholders. -->

<!--<field name="companies" separator="|"/>-->

<field name="companies" type="string" multiValued="true"/>

<!--<field name="locations" separator="|"/>-->

<field name="locations" type="string" multiValued="true"/>

<!--<field name="personnames" separator=";"/>-->

<field name="personnames" type="string" multiValued="true"/>

<!--<field name="concepts" separator="|" element-name="docvector\_navigator"/>-->

<field name="concepts" type="string" multiValued="true"/>

<!--<field name="emails" separator="|"/>-->

<field name="emails" type="string" multiValued="true"/>

<!--<field name="taxonomy" separator="#"/>-->

<field name="taxonomy" type="string" multiValued="true"/>

<!-- To be deleted -->

<!--<field name="emaildomain" element-name="email.domain"/>-->

<!--<field name="emaildomain" type="string"/>-->

<!-- All Collections -->

<!--<field name="emaildomains" separator=";"/>-->

<field name="emaildomains" type="string" multiValued="true"/>

<!-- end nested document -->

<!--Fields to store ngrams with different values of n-->

<!--<field name="addressngramfield2" index="yes" separator=";"/>-->

<field name="addressngram" type="ngram"/>

<!-- end nested document -->

<!--Fields to store ngrams for subjectname with different values of n-->

<!--<field name="subjectngramfield2" index="yes" separator=";"/>-->

<field name="subjectngram" type="ngram"/>

<!-- Military Fields -->

<!-- CIS, IDT -->

<!--<field name="militaryinstallationcode" index="no" result="dynamic" fallback-ref="militaryinstallationcode"/>-->

<field name="militaryinstallationcode" type="string" indexed="false"/>

<!-- CIS, IDT -->

<!--<field name="militarypaygradecode" index="no" result="dynamic" fallback-ref="militarypaygradecode"/>-->

<field name="militarypaygradecode" type="string" indexed="false"/>

<!-- CIS, IDT -->

<!--<field name="militarystatuscode" index="no" result="dynamic" fallback-ref="subjectnamevariations"/>-->

<field name="militarystatuscode" type="string" indexed="false"/>

<!-- CIS, IDT -->

<!--<field name="militarybranch" index="yes" result="dynamic" fallback-ref="militarybranch" separator="|"/>-->

<field name="militarybranch" type="string"/>

<!-- CIS, IDT -->

<!--<field name="militarymember" index="yes" result="dynamic" fallback-ref="militarymember"/>-->

<field name="militarymember" type="string"/>

<!-- CIS, IDT -->

<!--<field name="consumercountyname" index="yes" result="dynamic" fallback-ref="consumercountyname" separator="|"/>-->

<field name="consumercountyname" type="string"/>

<!-- CIS, IDT -->

<!--<field name="consumercountystatecode" index="yes" result="dynamic" fallback-ref="consumercountystatecode" separator="|"/>-->

<field name="consumercountystatecode" type="string"/>

<!-- CIS, IDT -->

<!--<field name="consumercountynameextended" index="yes" result="dynamic" fallback-ref="consumercountynameextended" separator="|"/>-->

<field name="consumercountynameextended" type="string"/>

<!-- CIS, IDT -->

<!--<field name="companycountyname" index="yes" result="dynamic" fallback-ref="companycountyname" separator="|"/>-->

<field name="companycountyname" type="string"/>

<!-- CIS, IDT -->

<!--<field name="companycountystatecode" index="yes" result="dynamic" fallback-ref="companycountystatecode" separator="|"/>-->

<field name="companycountystatecode" type="string"/>

<!-- CIS, IDT -->

<!--<field name="companycountynameextended" index="yes" result="dynamic" fallback-ref="companycountynameextended" separator="|"/>-->

<field name="companycountynameextended" type="string"/>

<!-- Cross Border CI -->

<!--<field name="crossborderflag" index="yes"/>-->

<field name="crossborderflag" type="string"/>

<!-- DNC -->

<!--<field name="subjectmatter" result="dynamic" fallback-ref="subjectmatter"/>-->

<field name="subjectmatter" type="string"/>

<!-- Entity and Phrases extraction -->

<!--<field name="extractedphrases" index="yes" separator="|"/>-->

<field name="extractedphrases" type="string" multiValued="true"/>

<!--<field name="extractedentities" index="yes" separator="|"/>-->

<field name="extractedentities" type="string" multiValued="true"/>

<!--<field name="extphraseentity" separator="|" result="dynamic" fallback-ref="extphraseentity"/>-->

<field name="extphraseentity" type="string" multiValued="true"/>

<!--<field name="extnamedentity" separator="|" result="dynamic" fallback-ref="extnamedentity"/>-->

<field name="extnamedentity" type="string" multiValued="true"/>

<!--<field name="extcompanies" separator="|" result="dynamic" fallback-ref="extcompanies"/>-->

<field name="extcompanies" type="string" multiValued="true"/>

<!--<field name="extamountpaid" type="float" result="dynamic" fallback-ref="extamountpaid" sort="yes"/>-->

<field name="extamountpaid" type="float"/>

<!--<field name="extlocations" separator="|" result="dynamic" fallback-ref="extlocations"/>-->

<field name="extlocations" type="string" multiValued="true"/>

<!--<field name="exturls" separator="|" result="dynamic" fallback-ref="exturls"/>-->

<field name="exturls" type="string" multiValued="true"/>

<!--<field name="yearquarter" type="int32" fallback-ref="yearquarter"/>-->

<field name="yearquarter" type="int"/>

<!--<field name="createdyear" type="int32" fallback-ref="createdyear"/>-->

<field name="createdyear" type="int"/>

<!--<field name="extphrasesmultiword" separator="|" result="dynamic" fallback-ref="extphrasesmultiword"/>-->

<field name="extphrasesmultiword" type="string" multiValued="true"/>

<!--<field name="amountinvolved" type="float" result="dynamic" fallback-ref="amountinvolved"/>-->

<field name="amountinvolved" type="float"/>

<!--<field name="extpaymentmethod" separator="|" result="dynamic" fallback-ref="extpaymentmethod"/>-->

<field name="extpaymentmethod" type="string" multiValued="true"/>

<!-- todo Geosearch fields. -->

<!--<field name="consumerlatitude" type="geo"/>

<field name="consumerlongitude" type="geo"/>

<field name="companylatitude" type="geo"/>

<field name="companylongitude" type="geo"/>-->

<!-- CIS -->

<!--<field name="consumerlatitudefloat" type="float" result="dynamic" fallback-ref="consumerlatitudefloat"/>-->

<field name="consumerlatitudefloat" type="float"/>

<!--<field name="consumerlongitudefloat" type="float" result="dynamic" fallback-ref="consumerlongitudefloat"/>-->

<field name="consumerlongitudefloat" type="float"/>

<!--<field name="companylatitudefloat" type="float" result="dynamic" fallback-ref="companylatitudefloat"/>-->

<field name="companylatitudefloat" type="float"/>

<!--<field name="companylongitudefloat" type="float" result="dynamic" fallback-ref="companylongitudefloat"/>-->

<field name="companylongitudefloat" type="float"/>

<!--additional generic fields -->

<!--<field name="genericstr1" result="dynamic" fallback-ref="genericstr1"/>-->

<field name="genericstr1" type="string"/>

<!--<field name="genericstr2" result="dynamic" fallback-ref="genericstr2"/>-->

<field name="genericstr2" type="string"/>

<!--<field name="genericint1" type="int32" fullsort="yes" result="dynamic" fallback-ref="genericint1"/>-->

<field name="genericint1" type="int"/>

<!--<field name="genericint2" type="int32" fullsort="yes" result="dynamic" fallback-ref="genericint2"/>-->

<field name="genericint2" type="int"/>

<!--<field name="genericdt1" type="datetime" fullsort="yes" result="dynamic" fallback-ref="genericdt1"/>-->

<field name="genericdt1" type="date"/>

<!--<field name="genericdt2" type="datetime" fullsort="yes" result="dynamic" fallback-ref="genericdt2"/>-->

<field name="genericdt2" type="date"/>

<!-- solr system fields -->

<field name="\_root\_" type="string" stored="false"/>

<field name="\_version\_" type="long"/>

</fields>

<uniqueKey>documentid</uniqueKey>

<!--<defaultSearchField>Body</defaultSearchField>-->

<solrQueryParser defaultOperator="OR"/>

<copyField source="concepts" dest="docvector\_navigator"/>

<!--<copyField source="emaildomain" dest="email.domain"/>-->

<copyField source="primarycompanycity" dest="content"/>

<copyField source="primarycompanystate" dest="content"/>

<copyField source="associatedcompanycity" dest="content"/>

<copyField source="associatedcompanystate" dest="content"/>

<copyField source="consumercity" dest="content"/>

<copyField source="consumerstate" dest="content"/>

<copyField source="consumerzip" dest="content"/>

<copyField source="prodservicedesc" dest="content"/>

<copyField source="comments" dest="content"/>

<copyField source="referencenumber" dest="content"/>

<copyField source="organization" dest="content"/>

<copyField source="creditbureauname" dest="content"/>

<copyField source="complainingcompany" dest="content"/>

<copyField source="consumerhomenumber" dest="content"/>

<copyField source="consumercellnumber" dest="content"/>

<copyField source="primarycompanyemail" dest="content"/>

<copyField source="associatedcompanyemail" dest="content"/>

<copyField source="primarycompanyphonenumberqs" dest="content"/>

<copyField source="primarycompanyname" dest="content"/>

<copyField source="associatedcompanyname" dest="content"/>

<copyField source="consumerfname" dest="content"/>

<copyField source="consumerlname" dest="content"/>

<copyField source="suspectfname" dest="content"/>

<copyField source="suspectlname" dest="content"/>

<copyField source="consumeraddr1" dest="content"/>

<copyField source="consumeraddr2" dest="content"/>

<copyField source="primarycompanyaddr1" dest="content"/>

<copyField source="associatedcompanyaddr1" dest="content"/>

<copyField source="primarycompanycity" dest="content"/>

<copyField source="militarybranch" dest="content"/>

<copyField source="soldierstatus" dest="content"/>

<copyField source="soldierstation" dest="content"/>

<copyField source="primarycompanycity" dest="contentwithbasename"/>

<copyField source="primarycompanystate" dest="contentwithbasename"/>

<copyField source="associatedcompanycity" dest="contentwithbasename"/>

<copyField source="associatedcompanystate" dest="contentwithbasename"/>

<copyField source="consumercity" dest="contentwithbasename"/>

<copyField source="consumerstate" dest="contentwithbasename"/>

<copyField source="consumerzip" dest="contentwithbasename"/>

<copyField source="prodservicedesc" dest="contentwithbasename"/>

<copyField source="comments" dest="contentwithbasename"/>

<copyField source="referencenumber" dest="contentwithbasename"/>

<copyField source="organization" dest="contentwithbasename"/>

<copyField source="consumercellnumber" dest="contentwithbasename"/>

<copyField source="creditbureauname" dest="contentwithbasename"/>

<copyField source="complainingcompany" dest="contentwithbasename"/>

<copyField source="primarycompanyphonenumberqs" dest="contentwithbasename"/>

<copyField source="consumerhomenumber" dest="contentwithbasename"/>

<copyField source="primarycompanyemail" dest="contentwithbasename"/>

<copyField source="associatedcompanyemail" dest="contentwithbasename"/>

<copyField source="subjectname" dest="contentwithbasename"/>

<copyField source="primarycompanyname" dest="contentwithbasename"/>

<copyField source="associatedcompanyname" dest="contentwithbasename"/>

<copyField source="subjectbasename" dest="contentwithbasename"/>

<copyField source="consumerfname" dest="contentwithbasename"/>

<copyField source="consumerlname" dest="contentwithbasename"/>

<copyField source="suspectfname" dest="contentwithbasename"/>

<copyField source="suspectlname" dest="contentwithbasename"/>

<copyField source="militarybranch" dest="contentwithbasename"/>

<copyField source="soldierstatus" dest="contentwithbasename"/>

<copyField source="soldierstation" dest="contentwithbasename"/>

<copyField source="consumeraddr1" dest="contentwithbasename"/>

<copyField source="consumeraddr2" dest="contentwithbasename"/>

<copyField source="primarycompanyaddr1" dest="contentwithbasename"/>

<copyField source="associatedcompanyaddr1" dest="contentwithbasename"/>

<copyField source="primarycompanycity" dest="contentwithbasename"/>

<copyField source="suspectfname" dest="cfalertsidt"/>

<copyField source="suspectlname" dest="cfalertsidt"/>

<copyField source="consumerfname" dest="cfalertsidt"/>

<copyField source="consumerlname" dest="cfalertsidt"/>

<copyField source="associatedcompanycity" dest="cfalertsidt"/>

<copyField source="associatedcompanystate" dest="cfalertsidt"/>

<copyField source="consumercity" dest="cfalertsidt"/>

<copyField source="consumerstate" dest="cfalertsidt"/>

<copyField source="associatedcompanyaddr1" dest="cfalertsidt"/>

<copyField source="consumeraddr1" dest="cfalertsidt"/>

<copyField source="soldierstation" dest="cfalertsidt"/>

<copyField source="associatedcompanycountry" dest="cfalertsidt"/>

<copyField source="consumercountry" dest="cfalertsidt"/>

<copyField source="associatedcompanyzip" dest="cfalertsidt"/>

<copyField source="consumerzip" dest="cfalertsidt"/>

<copyField source="associatedcompanyemail" dest="cfalertsidt"/>

<copyField source="associatedcompanyurl" dest="cfalertsidt"/>

<copyField source="consumeremail" dest="cfalertsidt"/>

<copyField source="associatedcompanycountrycode" dest="cfalertsidt"/>

<copyField source="associatedcompanyareacode" dest="cfalertsidt"/>

<copyField source="associatedcompanyphonenumber" dest="cfalertsidt"/>

<copyField source="primarycompanycity" dest="cfalertscis"/>

<copyField source="primarycompanystate" dest="cfalertscis"/>

<copyField source="primarycompanyaddr1" dest="cfalertscis"/>

<copyField source="primarycompanycountry" dest="cfalertscis"/>

<copyField source="primarycompanyzip" dest="cfalertscis"/>

<copyField source="primarycompanyemail" dest="cfalertscis"/>

<copyField source="primarycompanyurl" dest="cfalertscis"/>

<copyField source="primarycompanyname" dest="cfalertscis"/>

<copyField source="alerttypeofbusines" dest="cfalertscis"/>

<copyField source="soldierstation" dest="cfalertscis"/>

<copyField source="primarycompanycountrycode" dest="cfalertscis"/>

<copyField source="primarycompanyareacode" dest="cfalertscis"/>

<copyField source="primarycompanyphonenumber" dest="cfalertscis"/>

<copyField source="title" dest="cftutorials"/>

<copyField source="body" dest="cftutorials"/>

<copyField source="description" dest="cftutorials"/>

<copyField source="urlkeywords" dest="cftutorials"/>

<copyField source="keywords" dest="cftutorials"/>

<copyField source="anchortext" dest="cftutorials"/>

<copyField source="faqquestion" dest="cffaq"/>

<copyField source="faqanswer" dest="cffaq"/>

<types>

<!-- field type definitions. The "name" attribute is

just a label to be used by field definitions. The "class"

attribute and any other attributes determine the real

behavior of the fieldType.

Class names starting with "solr" refer to java classes in a

standard package such as org.apache.solr.analysis

-->

<fieldType name="ngram" class="solr.TextField" sortMissingLast="true">

<analyzer>

<tokenizer class="solr.NGramTokenizerFactory" minGramSize="6" maxGramSize="14"/>

</analyzer>

</fieldType>

<!-- The StrField type is not analyzed, but indexed/stored verbatim.

It supports doc values but in that case the field needs to be

single-valued and either required or have a default value.

-->

<fieldType name="string" class="solr.StrField" sortMissingLast="true"/>

<!-- boolean type: "true" or "false" -->

<fieldType name="boolean" class="solr.BoolField" sortMissingLast="true"/>

<!-- sortMissingLast and sortMissingFirst attributes are optional attributes are

currently supported on types that are sorted internally as strings

and on numeric types.

This includes "string","boolean", and, as of 3.5 (and 4.x),

int, float, long, date, double, including the "Trie" variants.

- If sortMissingLast="true", then a sort on this field will cause documents

without the field to come after documents with the field,

regardless of the requested sort order (asc or desc).

- If sortMissingFirst="true", then a sort on this field will cause documents

without the field to come before documents with the field,

regardless of the requested sort order.

- If sortMissingLast="false" and sortMissingFirst="false" (the default),

then default lucene sorting will be used which places docs without the

field first in an ascending sort and last in a descending sort.

-->

<!--

Default numeric field types. For faster range queries, consider the tint/tfloat/tlong/tdouble types.

These fields support doc values, but they require the field to be

single-valued and either be required or have a default value.

-->

<fieldType name="int" class="solr.TrieIntField" precisionStep="0" positionIncrementGap="0"/>

<fieldType name="float" class="solr.TrieFloatField" precisionStep="0" positionIncrementGap="0"/>

<fieldType name="long" class="solr.TrieLongField" precisionStep="0" positionIncrementGap="0"/>

<fieldType name="double" class="solr.TrieDoubleField" precisionStep="0" positionIncrementGap="0"/>

<!--

Numeric field types that index each value at various levels of precision

to accelerate range queries when the number of values between the range

endpoints is large. See the javadoc for NumericRangeQuery for internal

implementation details.

Smaller precisionStep values (specified in bits) will lead to more tokens

indexed per value, slightly larger index size, and faster range queries.

A precisionStep of 0 disables indexing at different precision levels.

-->

<fieldType name="tint" class="solr.TrieIntField" precisionStep="8" positionIncrementGap="0"/>

<fieldType name="tfloat" class="solr.TrieFloatField" precisionStep="8" positionIncrementGap="0"/>

<fieldType name="tlong" class="solr.TrieLongField" precisionStep="8" positionIncrementGap="0"/>

<fieldType name="tdouble" class="solr.TrieDoubleField" precisionStep="8" positionIncrementGap="0"/>

<!-- The format for this date field is of the form 1995-12-31T23:59:59Z, and

is a more restricted form of the canonical representation of dateTime

http://www.w3.org/TR/xmlschema-2/#dateTime

The trailing "Z" designates UTC time and is mandatory.

Optional fractional seconds are allowed: 1995-12-31T23:59:59.999Z

All other components are mandatory.

Expressions can also be used to denote calculations that should be

performed relative to "NOW" to determine the value, ie...

NOW/HOUR

... Round to the start of the current hour

NOW-1DAY

... Exactly 1 day prior to now

NOW/DAY+6MONTHS+3DAYS

... 6 months and 3 days in the future from the start of

the current day

Consult the DateField javadocs for more information.

Note: For faster range queries, consider the tdate type

-->

<fieldType name="date" class="solr.TrieDateField" precisionStep="0" positionIncrementGap="0"/>

<!-- A Trie based date field for faster date range queries and date faceting. -->

<fieldType name="tdate" class="solr.TrieDateField" precisionStep="6" positionIncrementGap="0"/>

<!--Binary data type. The data should be sent/retrieved in as Base64 encoded Strings -->

<fieldtype name="binary" class="solr.BinaryField"/>

<!--

Note:

These should only be used for compatibility with existing indexes (created with lucene or older Solr versions).

Use Trie based fields instead. As of Solr 3.5 and 4.x, Trie based fields support sortMissingFirst/Last

Plain numeric field types that store and index the text

value verbatim (and hence don't correctly support range queries, since the

lexicographic ordering isn't equal to the numeric ordering)

-->

<fieldType name="pint" class="solr.IntField"/>

<fieldType name="plong" class="solr.LongField"/>

<fieldType name="pfloat" class="solr.FloatField"/>

<fieldType name="pdouble" class="solr.DoubleField"/>

<fieldType name="pdate" class="solr.DateField" sortMissingLast="true"/>

<!-- The "RandomSortField" is not used to store or search any

data. You can declare fields of this type it in your schema

to generate pseudo-random orderings of your docs for sorting

or function purposes. The ordering is generated based on the field

name and the version of the index. As long as the index version

remains unchanged, and the same field name is reused,

the ordering of the docs will be consistent.

If you want different psuedo-random orderings of documents,

for the same version of the index, use a dynamicField and

change the field name in the request.

-->

<fieldType name="random" class="solr.RandomSortField" indexed="true"/>

<!-- solr.TextField allows the specification of custom text analyzers

specified as a tokenizer and a list of token filters. Different

analyzers may be specified for indexing and querying.

The optional positionIncrementGap puts space between multiple fields of

this type on the same document, with the purpose of preventing false phrase

matching across fields.

For more info on customizing your analyzer chain, please see

http://wiki.apache.org/solr/AnalyzersTokenizersTokenFilters

-->

<!-- One can also specify an existing Analyzer class that has a

default constructor via the class attribute on the analyzer element.

Example:

<fieldType name="text\_greek" class="solr.TextField">

<analyzer class="org.apache.lucene.analysis.el.GreekAnalyzer"/>

</fieldType>

-->

<!-- A text field that only splits on whitespace for exact matching of words -->

<fieldType name="text\_ws" class="solr.TextField" positionIncrementGap="100">

<analyzer>

<tokenizer class="solr.WhitespaceTokenizerFactory"/>

</analyzer>

</fieldType>

<!-- A general text field that has reasonable, generic

cross-language defaults: it tokenizes with StandardTokenizer,

removes stop words from case-insensitive "stopwords.txt"

(empty by default), and down cases. At query time only, it

also applies synonyms. -->

<fieldType name="text\_general" class="solr.TextField" positionIncrementGap="100">

<analyzer type="index">

<tokenizer class="solr.StandardTokenizerFactory"/>

<filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt"/>

<!-- in this example, we will only use synonyms at query time

<filter class="solr.SynonymFilterFactory" synonyms="index\_synonyms.txt" ignoreCase="true" expand="false"/>

-->

<filter class="solr.LowerCaseFilterFactory"/>

</analyzer>

<analyzer type="query">

<tokenizer class="solr.StandardTokenizerFactory"/>

<filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt"/>

<filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true" expand="true"/>

<filter class="solr.LowerCaseFilterFactory"/>

</analyzer>

</fieldType>

<!-- A text field with defaults appropriate for English: it

tokenizes with StandardTokenizer, removes English stop words

(lang/stopwords\_en.txt), down cases, protects words from protwords.txt, and

finally applies Porter's stemming. The query time analyzer

also applies synonyms from synonyms.txt. -->

<fieldType name="text\_en" class="solr.TextField" positionIncrementGap="100">

<analyzer type="index">

<tokenizer class="solr.StandardTokenizerFactory"/>

<!-- in this example, we will only use synonyms at query time

<filter class="solr.SynonymFilterFactory" synonyms="index\_synonyms.txt" ignoreCase="true" expand="false"/>

-->

<!-- Case insensitive stop word removal.

-->

<filter class="solr.StopFilterFactory"

ignoreCase="true"

words="lang/stopwords\_en.txt"

/>

<filter class="solr.LowerCaseFilterFactory"/>

<filter class="solr.EnglishPossessiveFilterFactory"/>

<filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>

<!-- Optionally you may want to use this less aggressive stemmer instead of PorterStemFilterFactory:

<filter class="solr.EnglishMinimalStemFilterFactory"/>

-->

<filter class="solr.PorterStemFilterFactory"/>

</analyzer>

<analyzer type="query">

<tokenizer class="solr.StandardTokenizerFactory"/>

<filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true" expand="true"/>

<filter class="solr.StopFilterFactory"

ignoreCase="true"

words="lang/stopwords\_en.txt"

/>

<filter class="solr.LowerCaseFilterFactory"/>

<filter class="solr.EnglishPossessiveFilterFactory"/>

<filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>

<!-- Optionally you may want to use this less aggressive stemmer instead of PorterStemFilterFactory:

<filter class="solr.EnglishMinimalStemFilterFactory"/>

-->

<filter class="solr.PorterStemFilterFactory"/>

</analyzer>

</fieldType>

<!-- A text field with defaults appropriate for English, plus

aggressive word-splitting and autophrase features enabled.

This field is just like text\_en, except it adds

WordDelimiterFilter to enable splitting and matching of

words on case-change, alpha numeric boundaries, and

non-alphanumeric chars. This means certain compound word

cases will work, for example query "wi fi" will match

document "WiFi" or "wi-fi".

-->

<fieldType name="text\_en\_splitting" class="solr.TextField" positionIncrementGap="100"

autoGeneratePhraseQueries="true">

<analyzer type="index">

<tokenizer class="solr.WhitespaceTokenizerFactory"/>

<!-- in this example, we will only use synonyms at query time

<filter class="solr.SynonymFilterFactory" synonyms="index\_synonyms.txt" ignoreCase="true" expand="false"/>

-->

<!-- Case insensitive stop word removal.

-->

<filter class="solr.StopFilterFactory"

ignoreCase="true"

words="lang/stopwords\_en.txt"

/>

<filter class="solr.WordDelimiterFilterFactory" generateWordParts="1" generateNumberParts="1"

catenateWords="1" catenateNumbers="1" catenateAll="0" splitOnCaseChange="1"/>

<filter class="solr.LowerCaseFilterFactory"/>

<filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>

<filter class="solr.PorterStemFilterFactory"/>

</analyzer>

<analyzer type="query">

<tokenizer class="solr.WhitespaceTokenizerFactory"/>

<filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true" expand="true"/>

<filter class="solr.StopFilterFactory"

ignoreCase="true"

words="lang/stopwords\_en.txt"

/>

<filter class="solr.WordDelimiterFilterFactory" generateWordParts="1" generateNumberParts="1"

catenateWords="0" catenateNumbers="0" catenateAll="0" splitOnCaseChange="1"/>

<filter class="solr.LowerCaseFilterFactory"/>

<filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>

<filter class="solr.PorterStemFilterFactory"/>

</analyzer>

</fieldType>

<!-- Less flexible matching, but less false matches. Probably not ideal for product names,

but may be good for SKUs. Can insert dashes in the wrong place and still match. -->

<fieldType name="text\_en\_splitting\_tight" class="solr.TextField" positionIncrementGap="100"

autoGeneratePhraseQueries="true">

<analyzer>

<tokenizer class="solr.WhitespaceTokenizerFactory"/>

<filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true" expand="false"/>

<filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords\_en.txt"/>

<filter class="solr.WordDelimiterFilterFactory" generateWordParts="0" generateNumberParts="0"

catenateWords="1" catenateNumbers="1" catenateAll="0"/>

<filter class="solr.LowerCaseFilterFactory"/>

<filter class="solr.KeywordMarkerFilterFactory" protected="protwords.txt"/>

<filter class="solr.EnglishMinimalStemFilterFactory"/>

<!-- this filter can remove any duplicate tokens that appear at the same position - sometimes

possible with WordDelimiterFilter in conjuncton with stemming. -->

<filter class="solr.RemoveDuplicatesTokenFilterFactory"/>

</analyzer>

</fieldType>

<!-- Just like text\_general except it reverses the characters of

each token, to enable more efficient leading wildcard queries. -->

<fieldType name="text\_general\_rev" class="solr.TextField" positionIncrementGap="100">

<analyzer type="index">

<tokenizer class="solr.StandardTokenizerFactory"/>

<filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt"/>

<filter class="solr.LowerCaseFilterFactory"/>

<filter class="solr.ReversedWildcardFilterFactory" withOriginal="true"

maxPosAsterisk="3" maxPosQuestion="2" maxFractionAsterisk="0.33"/>

</analyzer>

<analyzer type="query">

<tokenizer class="solr.StandardTokenizerFactory"/>

<filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true" expand="true"/>

<filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt"/>

<filter class="solr.LowerCaseFilterFactory"/>

</analyzer>

</fieldType>

<!-- charFilter + WhitespaceTokenizer -->

<!--

<fieldType name="text\_char\_norm" class="solr.TextField" positionIncrementGap="100" >

<analyzer>

<charFilter class="solr.MappingCharFilterFactory" mapping="mapping-ISOLatin1Accent.txt"/>

<tokenizer class="solr.WhitespaceTokenizerFactory"/>

</analyzer>

</fieldType>

-->

<!-- This is an example of using the KeywordTokenizer along

With various TokenFilterFactories to produce a sortable field

that does not include some properties of the source text

-->

<fieldType name="alphaOnlySort" class="solr.TextField" sortMissingLast="true" omitNorms="true">

<analyzer>

<!-- KeywordTokenizer does no actual tokenizing, so the entire

input string is preserved as a single token

-->

<tokenizer class="solr.KeywordTokenizerFactory"/>

<!-- The LowerCase TokenFilter does what you expect, which can be

when you want your sorting to be case insensitive

-->

<filter class="solr.LowerCaseFilterFactory"/>

<!-- The TrimFilter removes any leading or trailing whitespace -->

<filter class="solr.TrimFilterFactory"/>

<!-- The PatternReplaceFilter gives you the flexibility to use

Java Regular expression to replace any sequence of characters

matching a pattern with an arbitrary replacement string,

which may include back references to portions of the original

string matched by the pattern.

See the Java Regular Expression documentation for more

information on pattern and replacement string syntax.

http://java.sun.com/j2se/1.6.0/docs/api/java/util/regex/package-summary.html

-->

<filter class="solr.PatternReplaceFilterFactory"

pattern="([^a-z])" replacement="" replace="all"

/>

</analyzer>

</fieldType>

<fieldtype name="phonetic" stored="false" indexed="true" class="solr.TextField">

<analyzer>

<tokenizer class="solr.StandardTokenizerFactory"/>

<filter class="solr.DoubleMetaphoneFilterFactory" inject="false"/>

</analyzer>

</fieldtype>

<fieldtype name="payloads" stored="false" indexed="true" class="solr.TextField">

<analyzer>

<tokenizer class="solr.WhitespaceTokenizerFactory"/>

<!--

The DelimitedPayloadTokenFilter can put payloads on tokens... for example,

a token of "foo|1.4" would be indexed as "foo" with a payload of 1.4f

Attributes of the DelimitedPayloadTokenFilterFactory :

"delimiter" - a one character delimiter. Default is | (pipe)

"encoder" - how to encode the following value into a playload

float -> org.apache.lucene.analysis.payloads.FloatEncoder,

integer -> o.a.l.a.p.IntegerEncoder

identity -> o.a.l.a.p.IdentityEncoder

Fully Qualified class name implementing PayloadEncoder, Encoder must have a no arg constructor.

-->

<filter class="solr.DelimitedPayloadTokenFilterFactory" encoder="float"/>

</analyzer>

</fieldtype>

<!-- lowercases the entire field value, keeping it as a single token. -->

<fieldType name="lowercase" class="solr.TextField" positionIncrementGap="100">

<analyzer>

<tokenizer class="solr.KeywordTokenizerFactory"/>

<filter class="solr.LowerCaseFilterFactory"/>

</analyzer>

</fieldType>

<!--

Example of using PathHierarchyTokenizerFactory at index time, so

queries for paths match documents at that path, or in descendent paths

-->

<fieldType name="descendent\_path" class="solr.TextField">

<analyzer type="index">

<tokenizer class="solr.PathHierarchyTokenizerFactory" delimiter="/"/>

</analyzer>

<analyzer type="query">

<tokenizer class="solr.KeywordTokenizerFactory"/>

</analyzer>

</fieldType>

<!--

Example of using PathHierarchyTokenizerFactory at query time, so

queries for paths match documents at that path, or in ancestor paths

-->

<fieldType name="ancestor\_path" class="solr.TextField">

<analyzer type="index">

<tokenizer class="solr.KeywordTokenizerFactory"/>

</analyzer>

<analyzer type="query">

<tokenizer class="solr.PathHierarchyTokenizerFactory" delimiter="/"/>

</analyzer>

</fieldType>

<!-- since fields of this type are by default not stored or indexed,

any data added to them will be ignored outright. -->

<fieldtype name="ignored" stored="false" indexed="false" multiValued="true" class="solr.StrField"/>

<!-- This point type indexes the coordinates as separate fields (subFields)

If subFieldType is defined, it references a type, and a dynamic field

definition is created matching \*\_\_\_<typename>. Alternately, if

subFieldSuffix is defined, that is used to create the subFields.

Example: if subFieldType="double", then the coordinates would be

indexed in fields myloc\_0\_\_\_double,myloc\_1\_\_\_double.

Example: if subFieldSuffix="\_d" then the coordinates would be indexed

in fields myloc\_0\_d,myloc\_1\_d

The subFields are an implementation detail of the fieldType, and end

users normally should not need to know about them.

-->

<fieldType name="point" class="solr.PointType" dimension="2" subFieldSuffix="\_d"/>

<!-- A specialized field for geospatial search. If indexed, this fieldType must not be multivalued. -->

<fieldType name="location" class="solr.LatLonType" subFieldSuffix="\_coordinate"/>

<!-- An alternative geospatial field type new to Solr 4. It supports multiValued and polygon shapes.

For more information about this and other Spatial fields new to Solr 4, see:

http://wiki.apache.org/solr/SolrAdaptersForLuceneSpatial4

-->

<fieldType name="location\_rpt" class="solr.SpatialRecursivePrefixTreeFieldType"

geo="true" distErrPct="0.025" maxDistErr="0.000009" units="degrees"/>

<!-- Money/currency field type. See http://wiki.apache.org/solr/MoneyFieldType

Parameters:

defaultCurrency: Specifies the default currency if none specified. Defaults to "USD"

precisionStep: Specifies the precisionStep for the TrieLong field used for the amount

providerClass: Lets you plug in other exchange provider backend:

solr.FileExchangeRateProvider is the default and takes one parameter:

currencyConfig: name of an xml file holding exchange rates

solr.OpenExchangeRatesOrgProvider uses rates from openexchangerates.org:

ratesFileLocation: URL or path to rates JSON file (default latest.json on the web)

refreshInterval: Number of minutes between each rates fetch (default: 1440, min: 60)

-->

<fieldType name="currency" class="solr.CurrencyField" precisionStep="8" defaultCurrency="USD"

currencyConfig="currency.xml"/>

</types>

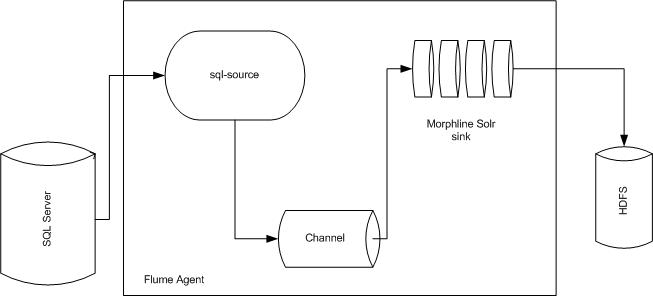
</schema>

## **Quick Search Queries**

|  |  |
| --- | --- |
| **Query Syntax** | **Expected Results** |
|  |  |
| content:"bank of america" | This one look for the whole pharse |
| content:bank\* | This one shall match any word with leading "bank" |
| content:"bank \*" | This one shall match any phrase with leading word "bank" |
| content:\*bank | Should match word or phrase with trailing "bank" . Observe some inconsistencies in SOLR |
| content:ban\*er | SOLR should return matches but does not. Needs further investigation |
| content:"credit card"~10 | Search for record where "credit" and "card" are within 10 words of each other |
|  | Ordered proximity search (as ONEAR in FAST) is not supported |
| createddate:[2011-01-01T00:00:00.000Z TO 2012-01-01T00:00:000Z] | Range search can be done for integer / datetime |
| content:("Credit" AND "Card") | Use of AND / OR operator |
| content:("Credit" OR "Card") | OR Operatot |
| content:traffic~ | Works only on single word |

# **Real Time Indexing using Flume & Morphlines**

Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of data. It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms. It uses a simple extensible data model that allows for online analytic application.



Flume has three components Source, Channel & Sink. A Flume agent is a (JVM) process that hosts the components through which events flow from an external source to the next destination (hop). A Flume source consumes events delivered to it by an external source like a SQL server. The external source sends events to Flume in a format that is recognized by the target Flume source. When a Flume source receives an event, it stores it into one or more channels. The channel is a passive store that keeps the event until it’s consumed by a Flume sink. The file channel is one example – it is backed by the local filesystem. The sink removes the event from the channel and puts it into an external repository like HDFS (via Flume HDFS sink) or forwards it to the Flume source of the next Flume agent (next hop) in the flow. The source and sink within the given agent run asynchronously with the events staged in the channel.

In our case sql-source acts as a flume source, system memory acts as a flume channel and solr-sink acts a flume sink. We need to configure our source, channel & sink in flume.conf located in /opt/cloudera/parcels/CDH-5.5.1-1.cdh5.5.1.p0.11/lib/flume-ng/conf.

## **Configure Flume sql-source**

* add flume-ng-sql-source-.jar into flume plugins dir folder

*$ mkdir -p $FLUME\_HOME/plugins.d/sql-source/lib $FLUME\_HOME/plugins.d/sql-source/libext*

*$ cp flume-ng-sql-source-0.8.jar $FLUME\_HOME/plugins.d/sql-source/lib*

* add SQL Server JDBC driver

*$ cp sqljdbc.jar $FLUME\_HOME/plugins.d/lib/sql-source/libext*

* configure flume.conf as follows
* sqlagent.sources = sql-source
* sqlagent.sinks = solrSink
* sqlagent.channels = memoryChannel
* sqlagent.sources.sql-source.type = org.keedio.flume.source.SQLSource
* sqlagent.sources.sql-source.connection.url =jdbc:sqlserver://ftcdevsql01.ppl.local:1433;database=cisbeta
* sqlagent.sources.sql-source.user = cis\_fast\_user
* sqlagent.sources.sql-source.password = cis4ftc
* sqlagent.sources.sql-source.table = dbo.FAST\_CISMain\_InitialLoad\_01192016
* sqlagent.sources.sql-source.max.rows = 1000
* sqlagent.sources.sql-source.columns.to.select = [Documentid],[RecordID],[Referencenum],convert(int,referencenumber) as intreferencenumber,[ContactType],[InternetRelatedFlag],[tsrrelatedflag],[LanguageName],[Source],[DNCFlag],[Comments],[CreatedDate],[CreatedBy],[UpdatedBy],[UpdatedDate],[OrgName],[AmountRequested],[AmountPaid],[PaymentMethod],[AgencyContact],[InitialContact],[InitialResponse],[ComplaintDate],[TransactionDate],[Topic],[ProdServiceDesc],[Statute],[Violation],[RepresentativeLastName],[RepresentativeFirstName],[RepresentativeTitle],[OrgID],[MoneyPaid],[Docacl],[quicksearchflag],[externalrefnumber],[complaintagainstcreditbureau],[complaintagainstcreditbureau45days],[acxml] as associatedcompany,[conxml] as consumer,[pcxml] as primarycompany,[CRADisputeResponded],[CRADisputeResolved],[MailImageFileName],[repxml] as agent,[MilitaryMember],[ComplaintResolutionFlag],[ComplaintResolutionText],[CrossBorderFlag]
* sqlagent.sources.sql-source.incremental.value = 0
* sqlagent.sources.sql-source.run.query.delay=10000
* sqlagent.sources.sql-source.status.file.path = /var/lib/flume
* sqlagent.sources.sql-source.status.file.name = sql-source.status
* agent.sources.sql-source.batch.size = 1000;
* sqlagent.sources.sql-source.channels = memoryChannel
* sqlagent.sinks.solrSink.type = org.apache.flume.sink.solr.morphline.MorphlineSolrSink
* sqlagent.sinks.solrSink.channel = memoryChannel
* sqlagent.sinks.solrSink.batchSize=1000
* sqlagent.sinks.solrSink.batchDurationMills=1000
* sqlagent.sinks.solrSink.morphlineFile=/opt/cloudera/parcels/CDH-5.5.1-1.cdh5.5.1.p0.11/lib/flume-ng/conf/morphlines.conf
* sqlagent.sinks.solrSink.morphlineId=morphline1
* sqlagent.channels.memoryChannel.type = memory
* sqlagent.channels.memoryChannel.capacity = 10000
* sqlagent.channels.memoryChannel.transactionCapacity = 10000

## **Morphline Configuration**

ZK\_HOST : "ftcdevcdh02.ppl.local:2181/solr"

SOLR\_LOCATOR : {

collection : ScopePOC

zkHost : ${ZK\_HOST}

}

morphlines : [

{

id : morphline1

importCommands : ["org.kitesdk.\*\*", "org.apache.solr.\*\*"]

commands : [

{

readCSV {

separator : ","

columns : [Documentid,RecordID,Referencenum,intreferencenumber,ContactType,InternetRelatedFlag,tsrrelatedflag,LanguageName,Source,DNCFlag,Comments,CreatedDate,CreatedBy,UpdatedBy,UpdatedDate,OrgName,AmountRequested,AmountPaid,PaymentMethod,AgencyContact,InitialContact,InitialResponse,ComplaintDate,TransactionDate,Topic,ProdServiceDesc,Statute,Violation,RepresentativeLastName,RepresentativeFirstName,RepresentativeTitle,OrgID,MoneyPaid,Docacl,quicksearchflag,externalrefnumber,complaintagainstcreditbureau,complaintagainstcreditbureau45days,associatedcompany,consumer,primarycompany,CRADisputeResponded,CRADisputeResolved,MailImageFileName,agent,MilitaryMember,ComplaintResolutionFlag,ComplaintResolutionText,CrossBorderFlag]

charset : UTF-8

addEmptyStrings : false

quoteChar : "\""

}

}

{

java {

imports : """

import org.kitesdk.morphline.api.Record;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.NodeList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMResult;

import javax.xml.transform.stream.StreamSource;

import java.io.ByteArrayInputStream;

"""

code : """

Record newRecord = new Record();

newRecord.put("recordid", record.getFirstValue("recordid"));

newRecord.put("referencenumber", record.getFirstValue("referencenumber"));

newRecord.put("intreferencenumber", record.getFirstValue("intreferencenumber"));

final String documentid = record.getFirstValue("documentid").toString();

newRecord.put("ContactType", record.getFirstValue("ContactType"));

newRecord.put("InternetRelatedFlag", record.getFirstValue("InternetRelatedFlag"));

newRecord.put("tsrrelatedflag", record.getFirstValue("tsrrelatedflag"));

newRecord.put("LanguageName", record.getFirstValue("LanguageName"));

newRecord.put("Source", record.getFirstValue("Source"));

newRecord.put("DNCFlag", record.getFirstValue("DNCFlag"));

newRecord.put("Comments", record.getFirstValue("Comments"));

newRecord.put("CreatedBy", record.getFirstValue("CreatedBy"));

newRecord.put("UpdatedDate", record.getFirstValue("UpdatedDate"));

newRecord.put("OrgName", record.getFirstValue("OrgName"));

newRecord.put("AmountRequested", record.getFirstValue("AmountRequested"));

newRecord.put("AmountPaid", record.getFirstValue("AmountPaid"));

newRecord.put("PaymentMethod", record.getFirstValue("PaymentMethod"));

newRecord.put("AgencyContact", record.getFirstValue("AgencyContact"));

newRecord.put("InitialResponse", record.getFirstValue("InitialResponse"));

newRecord.put("TransactionDate", record.getFirstValue("TransactionDate"));

newRecord.put("InitialContact", record.getFirstValue("InitialContact"));

newRecord.put("ComplaintDate", Crecord.getFirstValue("omplaintDate"));

newRecord.put("TransactionDate", record.getFirstValue("TransactionDate"));

newRecord.put("Topic", record.getFirstValue("Topic"));

newRecord.put("ProdServiceDesc", record.getFirstValue("ProdServiceDesc"));

newRecord.put("Statute", record.getFirstValue("Statute"));

newRecord.put("Violation", record.getFirstValue("Violation"));

newRecord.put("RepresentativeLastName", record.getFirstValue("RepresentativeLastName"));

newRecord.put("RepresentativeFirstName", record.getFirstValue("RepresentativeFirstName"));

newRecord.put("RepresentativeTitle", record.getFirstValue("RepresentativeTitle"));

newRecord.put("OrgID", record.getFirstValue("OrgID"));

newRecord.put("MoneyPaid", record.getFirstValue("MoneyPaid"));

newRecord.put("Docacl", record.getFirstValue("Docacl"));

newRecord.put("quicksearchflag", record.getFirstValue("quicksearchflag"));

newRecord.put("externalrefnumber", record.getFirstValue("externalrefnumber"));

newRecord.put("complaintagainstcreditbureau", record.getFirstValue("complaintagainstcreditbureau"));

newRecord.put("complaintagainstcreditbureau45days", record.getFirstValue("complaintagainstcreditbureau45days"));

newRecord.put("CRADisputeResponded", record.getFirstValue("CRADisputeResponded"));

newRecord.put("CRADisputeResolved", record.getFirstValue("CRADisputeResolved"));

newRecord.put("MailImageFileName", record.getFirstValue("MailImageFileName"));

newRecord.put("MilitaryMember", record.getFirstValue("MilitaryMember"));

newRecord.put("ComplaintResolutionFlag", record.getFirstValue("ComplaintResolutionFlag"));

newRecord.put("ComplaintResolutionText", record.getFirstValue("ComplaintResolutionText"));

newRecord.put("CrossBorderFlag", record.getFirstValue("CrossBorderFlag"));

newRecord.put("documentid", documentid);

newRecord.put("type\_s", "record");

newRecord.put("consumer",(String) record.getFirstValue("consumer"));

try {

String xmlString = (String) record.getFirstValue("primarycompany");

DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder builder = builderFactory.newDocumentBuilder();

Document document = builder.newDocument();

byte[] buffer = xmlString.getBytes();

ByteArrayInputStream stream = new ByteArrayInputStream(buffer);

StreamSource source = new StreamSource(stream);

DOMResult result = new DOMResult(document);

TransformerFactory tFactory = TransformerFactory.newInstance();

Transformer transformer = tFactory.newTransformer();

transformer.transform(source, result);

final Element documentElement = document.getDocumentElement();

final NodeList companyNodes = documentElement.getElementsByTagName("pc");

int size = companyNodes.getLength();

for (int i = 0; i < size; i++) {

Record childRecord = new Record();

childRecord.put("recordid", record.getFirstValue("recordid"));

childRecord.put("referencenumber", record.getFirstValue("referencenumber"));

childRecord.put("intreferencenumber", record.getFirstValue("intreferencenumber"));

childRecord.put("documentid", documentid + "C" + i);

childRecord.put("type\_s", "company");

final Element company = (Element) companyNodes.item(i);

childRecord.put("primarycompanyname", company.getElementsByTagName("name").item(0).getTextContent());

childRecord.put("primarycompanyaddr1", company.getElementsByTagName("address1").item(0).getTextContent());

childRecord.put("primarycompanyaddr2", company.getElementsByTagName("address2").item(0).getTextContent());

childRecord.put("primarycompanycity", company.getElementsByTagName("city").item(0).getTextContent());

childRecord.put("primarycompanystate", company.getElementsByTagName("state").item(0).getTextContent());

childRecord.put("primarycompanyzip", company.getElementsByTagName("zip").item(0).getTextContent());

childRecord.put("primarycompanycountry", company.getElementsByTagName("country").item(0).getTextContent());

childRecord.put("primarycompanyemail", company.getElementsByTagName("email").item(0).getTextContent());

childRecord.put("primarycompanyurl", company.getElementsByTagName("url").item(0).getTextContent());

childRecord.put("primarycompanycountrycode", company.getElementsByTagName("countrycode").item(0).getTextContent());

childRecord.put("primarycompanyareacode", company.getElementsByTagName("areacode").item(0).getTextContent());

childRecord.put("primarycompanyphonenumber", company.getElementsByTagName("phonenumber").item(0).getTextContent());

childRecord.put("primarycompanyphoneext", company.getElementsByTagName("phoneext").item(0).getTextContent());

childRecord.put("subjectid", company.getElementsByTagName("subjectid").item(0).getTextContent());

childRecord.put("subjectidtype", company.getElementsByTagName("subjectidtype").item(0).getTextContent());

childRecord.put("subjectidissuerstate", company.getElementsByTagName("subjectidissuerstate").item(0).getTextContent());

childRecord.put("subjectidissuercountry", company.getElementsByTagName("subjectidissuercountry").item(0).getTextContent());

newRecord.put("\_loadSolr\_childDocuments", childRecord);

}

} catch (Exception e) {

e.printStackTrace();

}

return child.process(newRecord);

"""

}

}

{

java {

imports : """

import org.kitesdk.morphline.api.Record;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.NodeList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMResult;

import javax.xml.transform.stream.StreamSource;

import java.io.ByteArrayInputStream;

"""

code : """

String xmlString = (String) record.getFirstValue("acs");

record.removeAll("acs");

final String documentid = record.getFirstValue("documentid").toString();

try {

DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder builder = builderFactory.newDocumentBuilder();

Document document = builder.newDocument();

byte[] buffer = xmlString.getBytes();

ByteArrayInputStream stream = new ByteArrayInputStream(buffer);

StreamSource source = new StreamSource(stream);

DOMResult result = new DOMResult(document);

TransformerFactory tFactory = TransformerFactory.newInstance();

Transformer transformer = tFactory.newTransformer();

transformer.transform(source, result);

final Element documentElement = document.getDocumentElement();

final NodeList consumerNodes = documentElement.getElementsByTagName("con");

int size = consumerNodes.getLength();

for (int i = 0; i < size; i++) {

Record childRecord = new Record();

childRecord.put("recordid", record.getFirstValue("recordid"));

childRecord.put("referencenumber", record.getFirstValue("referencenumber"));

childRecord.put("intreferencenumber", record.getFirstValue("intreferencenumber"));

childRecord.put("documentid", documentid + "Con" + i);

childRecord.put("type\_s", "acs");

final Element acs = (Element) consumerNodes.item(i);

childRecord.put("associatedcompanyname", acs.getElementsByTagName("name").item(0).getTextContent());

childRecord.put("associatedcompanybasename", acs.getElementsByTagName("basename").item(0).getTextContent());

childRecord.put("associatedcompanytype", acs.getElementsByTagName("type").item(0).getTextContent());

childRecord.put("associatedcompanyaddr1", acs.getElementsByTagName("address1").item(0).getTextContent());

childRecord.put("associatedcompanyaddr2", acs.getElementsByTagName("address2").item(0).getTextContent());

childRecord.put("associatedcompanycity", acs.getElementsByTagName("city").item(0).getTextContent());

childRecord.put("associatedcompanystate", acs.getElementsByTagName("state").item(0).getTextContent());

childRecord.put("associatedcompanyzip", acs.getElementsByTagName("zip").item(0).getTextContent());

childRecord.put("associatedcompanycountry", acs.getElementsByTagName("country").item(0).getTextContent());

childRecord.put("associatedcompanyemail", acs.getElementsByTagName("email").item(0).getTextContent());

childRecord.put("associatedcompanyurl", acs.getElementsByTagName("url").item(0).getTextContent());

childRecord.put("associatedcompanyareacode", acs.getElementsByTagName("areacode").item(0).getTextContent());

childRecord.put("associatedcompanyphonenumber", acs.getElementsByTagName("phonenumber").item(0).getTextContent());

childRecord.put("associatedcompanyphoneext", acs.getElementsByTagName("phoneext").item(0).getTextContent());

childRecord.put("representativelname", acs.getElementsByTagName("replname").item(0).getTextContent());

record.put("\_loadSolr\_childDocuments", childRecord);

}

} catch (Exception e) {

e.printStackTrace();

}

return child.process(record);

"""

}

}

{

java {

imports : """

import org.kitesdk.morphline.api.Record;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.NodeList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMResult;

import javax.xml.transform.stream.StreamSource;

import java.io.ByteArrayInputStream;

"""

code : """

String xmlString = (String) record.getFirstValue("consumer");

record.removeAll("consumer");

final String documentid = record.getFirstValue("documentid").toString();

try {

DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder builder = builderFactory.newDocumentBuilder();

Document document = builder.newDocument();

byte[] buffer = xmlString.getBytes();

ByteArrayInputStream stream = new ByteArrayInputStream(buffer);

StreamSource source = new StreamSource(stream);

DOMResult result = new DOMResult(document);

TransformerFactory tFactory = TransformerFactory.newInstance();

Transformer transformer = tFactory.newTransformer();

transformer.transform(source, result);

final Element documentElement = document.getDocumentElement();

final NodeList consumerNodes = documentElement.getElementsByTagName("con");

int size = consumerNodes.getLength();

for (int i = 0; i < size; i++) {

Record childRecord = new Record();

childRecord.put("recordid", record.getFirstValue("recordid"));

childRecord.put("referencenumber", record.getFirstValue("referencenumber"));

childRecord.put("intreferencenumber", record.getFirstValue("intreferencenumber"));

childRecord.put("documentid", documentid + "Con" + i);

childRecord.put("type\_s", "consumer");

final Element consumer = (Element) consumerNodes.item(i);

childRecord.put("consumerfname", consumer.getElementsByTagName("firtsname").item(0).getTextContent());

childRecord.put("consumerlname", consumer.getElementsByTagName("lastname").item(0).getTextContent());

childRecord.put("consumeremail", consumer.getElementsByTagName("email").item(0).getTextContent());

childRecord.put("consumeraddr1", consumer.getElementsByTagName("address1").item(0).getTextContent());

childRecord.put("consumeraddr2", consumer.getElementsByTagName("address2").item(0).getTextContent());

childRecord.put("consumercity", consumer.getElementsByTagName("city").item(0).getTextContent());

childRecord.put("consumerstate", consumer.getElementsByTagName("state").item(0).getTextContent());

childRecord.put("consumerzip", consumer.getElementsByTagName("zip").item(0).getTextContent());

childRecord.put("consumercountry", consumer.getElementsByTagName("country").item(0).getTextContent());

childRecord.put("consumerworkphone", consumer.getElementsByTagName("worknumber").item(0).getTextContent());

childRecord.put("consumercountrycode", consumer.getElementsByTagName("workcountrycode").item(0).getTextContent());

childRecord.put("consumerareacode", consumer.getElementsByTagName("workareacode").item(0).getTextContent());

childRecord.put("consumerworkphone", consumer.getElementsByTagName("workphonenumber").item(0).getTextContent());

childRecord.put("consumerworkphoneext", consumer.getElementsByTagName("workphoneextension").item(0).getTextContent());

childRecord.put("consumerfaxnumber", consumer.getElementsByTagName("faxnumber").item(0).getTextContent());

childRecord.put("consumerhomenumber", consumer.getElementsByTagName("homenumber").item(0).getTextContent());

childRecord.put("consumercellnumber", consumer.getElementsByTagName("cellnumber").item(0).getTextContent());

childRecord.put("consumeragerange", consumer.getElementsByTagName("agerange").item(0).getTextContent());

childRecord.put("militarybranch", consumer.getElementsByTagName("militarybranch").item(0).getTextContent());

childRecord.put("militarystatuscode", consumer.getElementsByTagName("militarystatus").item(0).getTextContent());

childRecord.put("militaryinstallationcode", consumer.getElementsByTagName("militaryinstallation").item(0).getTextContent());

childRecord.put("militarypaygradecode", consumer.getElementsByTagName("militarypaygrade").item(0).getTextContent());

record.put("\_loadSolr\_childDocuments", childRecord);

}

} catch (Exception e) {

e.printStackTrace();

}

return child.process(record);

"""

}

}

{

loadSolr {

solrLocator : ${SOLR\_LOCATOR}

}

}

{

java {

imports : """

import org.kitesdk.morphline.api.Record;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

"""

code : """

String documentid = (String) record.getFirstValue("documentid");

Connection conn = null;

try {

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

conn = DriverManager.getConnection("jdbc:sqlserver://ftcdevsql01.ppl.local:1433;database=cisbeta", "cis\_fast\_user", "cis4ftc");

Statement statement = conn.createStatement();

String queryString = "delete from dbo.FAST\_CISMain\_InitialLoad\_01192016 where documentid = '" + documentid + "'";

int value = statement.executeUpdate(queryString);

} catch (Exception e) {

e.printStackTrace();

} finally {

if (conn != null) try {

conn.close();

} catch (Exception e) {

e.printStackTrace();

}

}

return child.process(record);

"""

}

}

]

}

]

# **Initial Indexing**

Indexing logistics in SOLR can inherit a lot in terms of database configurations and processes from existing FAST indexing. We may continue to use the identical load tables from which Flume agent can crawl the records. In the SQL Source of the Flume we make use of a select statement similar to the one that follows-

Select documentid, documented as RecordId, referencenumber, convert(int, referencenumber) as intreferncenumber, pcxml as primarycompany, conxml as consumerxml, convert(varchar(33), createddate, 126) + ‘Z’ as createddate

Important to note that we do not need to roll up and make use of child entity fields such as Company, Consumer, fields (as multivalue fields). Scope XML fields can be used to capture and index information from child entity. As a result of that, we could do away with substantial DB processing during the loading of records.

Flume source is configured with a batch size properties. The source pulls in records up to the batch size in one go and push it to the channel. Sinks process these records and adds them to the index through Morphlines. Typically rate of processing at the Sink is slower than at the Source. Channel acts as buffer in this case.

Flume Sink sends back the status of completion to the channel and channel in turn communicates to the source for next batch pull. That way no additional support is needed from the database side. In fact no differential treatment is needed for incremental indexing. Same table can continue to serve as incremental load table.

# **Incremental Indexing**

Flume service once instantiated runs continuously and streams data as new data becomes available. Flume makes use of Incremental Column for keeping track of records already streamed and next record to be streamed. To take advantage of this feature, the incremental table is added with an Identity column (Id). The flume configuration that indicates the incremental column is as follows:

*# Increment column properties*

*agent1.sources.sql-source.incremental.column.name = id*

*# Increment value is from you want to start taking data from tables (0 will import entire table)*

*agent1.sources.sql-source.incremental.value = 0*

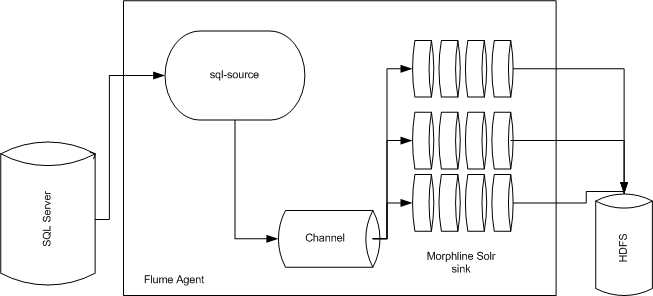
As Flume is done streaming all records from the incremental load table, it keeps a record of what the last Id that was crawled. It waits for further records with following criteria

*16/01/18 19:52:48 INFO source.SQLSource: Query: SELECT \* FROM FAST-CISMain-Updates WHERE id>100 ORDER BY id;*

As soon as another record gets added to the table, the streaming event gets fired and Flume executes the query to pull the record. This mechanism helps us achieving what we call Near Real Time indexing.

# **Performance Improvement**

Now flume is taking around 5 mins to create SOLR index with 10000 records. The main bottleneck in SOLR Morphline is the sink. Here sink is processing one record at a time, there is no way to configure a batch processing. But we can improve the performance by creating multiple sink. Flume support multiple sink reading from same channel and working on them. In this case our approach would be one SOLR-source writing to memory channel and then multiple SOLR-Sinks(say 3) are reading from the channel and creating SOLR index.



# **Appendix 1) Existing FAST Configurations**

## Quick Search Implementation

Existing FAST ESP implementation in CSN has 383 primary fields and 10 Scope fields. Some of these fields are copied into an index/composite field named “Content”. Quick Search is executed by searching this composite fields. Please refer the Appendix 1 for the complete index-profile.

## Composite Fields

There are multiple composite fields in the index-profile. Each composite field is used in specific query context. Each Composite field is made off up to 8 different field groups. Each field group is ranked which impacts the ranking. Primary Composite fields and their compositions are as follows:

### Composite Field – ***Content***

This composite field is used when all participating fields need to be searched with the original company name (excluding ***CompanyBaseName***).

<composite-field name="content" default="yes">

<field-ref-group name="fieldgroup1" level="1">

<field-ref name="primarycompanycity" field-separation-length="256"/> <!-- CIS, IDT, -->

<field-ref name="primarycompanystate" field-separation-length="256"/> <!-- CIS, IDT, DNC, -->

<field-ref name="associatedcompanycity" field-separation-length="256"/> <!-- CIS, IDT -->

<field-ref name="associatedcompanystate" field-separation-length="256"/> <!-- CIS, IDT -->

<field-ref name="consumercity" field-separation-length="256"/> <!-- CIS, IDT, DNC -->

<field-ref name="consumerstate" field-separation-length="256"/> <!-- CIS, IDT, DNC -->

<field-ref name="consumerzip" field-separation-length="256"/> <!-- CIS, IDT, DNC -->

</field-ref-group>

<field-ref-group name="fieldgroup2" level="2">

<field-ref name="prodservicedesc" field-separation-length="256"/> <!-- CIS, IDT, DNC, -->

<field-ref name="comments" field-separation-length="256"/> <!-- CIS, IDT, DNC -->

<field-ref name="referencenumber" field-separation-length="256"/> <!-- CIS, IDT, DNC, -->

<field-ref name="organization" field-separation-length="256"/> <!-- CIS, IDT, DNC, -->

<field-ref name="creditbureauname" field-separation-length="256"/> <!-- IDT Part II -->

<field-ref name="complainingcompany" field-separation-length="256"/> <!-- IDT Part II -->

</field-ref-group>

<field-ref-group name="fieldgroup3" level="3">

<field-ref name="consumerhomenumber" field-separation-length="256"/> <!-- IDT Part II-->

<field-ref name="consumercellnumber" field-separation-length="256"/> <!-- IDT Part II-->

<field-ref name="primarycompanyemail" field-separation-length="256"/> <!-- CIS, IDT -->

<field-ref name="associatedcompanyemail" field-separation-length="256"/> <!-- CIS -->

<field-ref name="primarycompanyphonenumberqs" field-separation-length="256"/> <!-- IDT Part II -->

</field-ref-group>

<field-ref-group name="fieldgroup4" level="4">

<field-ref name="primarycompanyname" field-separation-length="256"/> <!-- CIS, IDT, DNC, -->

<field-ref name="associatedcompanyname" field-separation-length="256"/> <!-- CIS, IDT -->

<field-ref name="consumerfname" field-separation-length="256"/> <!-- CIS, IDT, DNC -->

<field-ref name="consumerlname" field-separation-length="256"/> <!-- CIS, IDT, DNC -->

<field-ref name="suspectfname" field-separation-length="256"/> <!-- IDT -->

<field-ref name="suspectlname" field-separation-length="256"/> <!-- IDT -->

</field-ref-group>

<field-ref-group name="fieldgroup5" level="1">

<field-ref name="consumeraddr1" field-separation-length="256"/> <!-- CIS, DNC, -->

<field-ref name="consumeraddr2" field-separation-length="256"/> <!-- CIS, DNC, -->

<field-ref name="primarycompanyaddr1" field-separation-length="256"/> <!-- CIS -->

<field-ref name="associatedcompanyaddr1" field-separation-length="256"/> <!-- IDT -->

<field-ref name="primarycompanycity" field-separation-length="256"/> <!-- CIS, DNC, -->

</field-ref-group>

<field-ref-group name="fieldgroup6" level="2">

<field-ref name="militarybranch" field-separation-length="256"/> <!-- CIS -->

<field-ref name="soldierstatus" field-separation-length="256"/> <!-- CIS -->

<field-ref name="soldierstation" field-separation-length="256"/> <!-- CIS -->

</field-ref-group>

</composite-field>

### Composite Field – ContentWithBaseName

This composite field is used when all participating fields need to be searched including ***CompanyBaseName****.*

<composite-field name="contentwithbasename" default="no">

<field-ref-group name="fieldgroup1" level="1">

<field-ref name="primarycompanycity" field-separation-length="256"/> <!-- CIS, IDT, -->

<field-ref name="primarycompanystate" field-separation-length="256"/> <!-- CIS, IDT, DNC, -->

<field-ref name="associatedcompanycity" field-separation-length="256"/> <!-- CIS, IDT -->

<field-ref name="associatedcompanystate" field-separation-length="256"/> <!-- CIS, IDT -->

<field-ref name="consumercity" field-separation-length="256"/> <!-- CIS, IDT, DNC -->

<field-ref name="consumerstate" field-separation-length="256"/> <!-- CIS, IDT, DNC -->

<field-ref name="consumerzip" field-separation-length="256"/> <!-- CIS, IDT, DNC -->

</field-ref-group>

<field-ref-group name="fieldgroup2" level="2">

<field-ref name="prodservicedesc" field-separation-length="256"/> <!-- CIS, IDT, DNC, -->

<field-ref name="comments" field-separation-length="256"/> <!-- CIS, IDT, DNC -->

<field-ref name="referencenumber" field-separation-length="256"/> <!-- CIS, IDT, DNC, -->

<field-ref name="organization" field-separation-length="256"/> <!-- CIS, IDT, DNC, -->

<field-ref name="consumercellnumber" field-separation-length="256"/> <!-- IDT Part II-->

<field-ref name="creditbureauname" field-separation-length="256"/> <!-- IDT Part II -->

<field-ref name="complainingcompany" field-separation-length="256"/> <!-- IDT Part II -->

<field-ref name="primarycompanyphonenumberqs" field-separation-length="256"/> <!-- IDT Part II -->

</field-ref-group>

<field-ref-group name="fieldgroup3" level="3">

<field-ref name="consumerhomenumber" field-separation-length="256"/> <!-- IDT Part II-->

<field-ref name="primarycompanyemail" field-separation-length="256"/> <!-- CIS, IDT -->

<field-ref name="associatedcompanyemail" field-separation-length="256"/> <!-- CIS -->

<field-ref name="subjectname" field-separation-length="256"/> <!-- CIS, IDT, DNC, -->

</field-ref-group>

<field-ref-group name="fieldgroup4" level="4">

<field-ref name="primarycompanyname" field-separation-length="256"/> <!-- CIS, IDT, DNC, -->

<field-ref name="associatedcompanyname" field-separation-length="256"/> <!-- CIS, IDT -->

<field-ref name="subjectbasename" field-separation-length="256"/> <!-- CIS, IDT, DNC, -->

<field-ref name="consumerfname" field-separation-length="256"/> <!-- CIS, IDT, DNC -->

<field-ref name="consumerlname" field-separation-length="256"/> <!-- CIS, IDT, DNC -->

<field-ref name="suspectfname" field-separation-length="256"/> <!-- IDT -->

<field-ref name="suspectlname" field-separation-length="256"/> <!-- IDT -->

</field-ref-group>

<field-ref-group name="fieldgroup6" level="2">

<field-ref name="militarybranch" field-separation-length="256"/> <!-- CIS -->

<field-ref name="soldierstatus" field-separation-length="256"/> <!-- CIS -->

<field-ref name="soldierstation" field-separation-length="256"/> <!-- CIS -->

</field-ref-group>

<field-ref-group name="fieldgroup5" level="1">

<field-ref name="consumeraddr1" field-separation-length="256"/> <!-- CIS, DNC, -->

<field-ref name="consumeraddr2" field-separation-length="256"/> <!-- CIS, DNC, -->

<field-ref name="primarycompanyaddr1" field-separation-length="256"/> <!-- CIS -->

<field-ref name="associatedcompanyaddr1" field-separation-length="256"/> <!-- IDT -->

<field-ref name="primarycompanycity" field-separation-length="256"/> <!-- CIS, DNC, -->

</field-ref-group>

</composite-field>

### Composite Field – ***CFAlertsIDT***

This composite field is used to do perform quick search on IDT Alerts

<composite-field name="cfalertsidt" default="no" query-tokenize="auto">

<field-ref-group name="alertsgroup1" level="1">

<field-ref name="suspectfname" field-separation-length="256"/>

<field-ref name="suspectlname" field-separation-length="256"/>

<field-ref name="consumerfname" field-separation-length="256"/>

<field-ref name="consumerlname" field-separation-length="256"/>

</field-ref-group>

<field-ref-group name="alertsgroup2" level="2">

<field-ref name="associatedcompanycity" field-separation-length="256"/>

<field-ref name="associatedcompanystate" field-separation-length="256"/>

<field-ref name="consumercity" field-separation-length="256"/>

<field-ref name="consumerstate" field-separation-length="256"/>

</field-ref-group>

<field-ref-group name="alertsgroup3" level="3">

<field-ref name="associatedcompanyaddr1" field-separation-length="256"/>

<field-ref name="consumeraddr1" field-separation-length="256"/>

<field-ref name="soldierstation" field-separation-length="256"/>

</field-ref-group>

<field-ref-group name="alertsgroup4" level="3">

<field-ref name="associatedcompanycountry" field-separation-length="256"/>

<field-ref name="consumercountry" field-separation-length="256"/>

</field-ref-group>

<field-ref-group name="alertsgroup5" level="3">

<field-ref name="associatedcompanyzip" field-separation-length="256"/>

<field-ref name="consumerzip" field-separation-length="256"/>

</field-ref-group>

<field-ref-group name="alertsgroup6" level="3">

<field-ref name="associatedcompanyemail" field-separation-length="256"/>

<field-ref name="associatedcompanyurl" field-separation-length="256"/>

<field-ref name="consumeremail" field-separation-length="256"/>

</field-ref-group>

<field-ref-group name="alertsgroup7" level="3">

<field-ref name="associatedcompanycountrycode" field-separation-length="256"/>

<field-ref name="associatedcompanyareacode" field-separation-length="256"/>

<field-ref name="associatedcompanyphonenumber" field-separation-length="256"/>

</field-ref-group>

</composite-field>

### Composite Field – ***CFAlertsCIS***

This composite field is used to do perform quick search on CIS Alerts

<composite-field name="cfalertscis" default="no" query-tokenize="auto">

<field-ref-group name="alertscisgroup1" level="1">

<field-ref name="primarycompanycity" field-separation-length="256"/>

<field-ref name="primarycompanystate" field-separation-length="256"/>

<field-ref name="primarycompanyaddr1" field-separation-length="256"/>

<field-ref name="primarycompanycountry" field-separation-length="256"/>

<field-ref name="primarycompanyzip" field-separation-length="256"/>

<field-ref name="primarycompanyemail" field-separation-length="256"/>

<field-ref name="primarycompanyurl" field-separation-length="256"/>

</field-ref-group>

<field-ref-group name="alertscisgroup2" level="2">

<field-ref name="primarycompanyname" field-separation-length="256"/>

<field-ref name="alerttypeofbusines" field-separation-length="256"/>

<field-ref name="soldierstation" field-separation-length="256"/>

</field-ref-group>

<field-ref-group name="alertscisgroup3" level="3">

<field-ref name="primarycompanycountrycode" field-separation-length="256"/>

<field-ref name="primarycompanyareacode" field-separation-length="256"/>

<field-ref name="primarycompanyphonenumber" field-separation-length="256"/>

</field-ref-group>

</composite-field>

## Rank Profile

Existing CSN Implementation of FAST uses OOTB Rank Profile for primary search. However Alerts, FAQ and Tutorial searches uses their own Rank Profiles, which are minor updates of the OOTB rank profile. Existing CSN application did not really make use of the search relevancy. Rather results were sorted by created / updated date.

### Default Rank Profile

<rank-profile name="default" rank-model="default" default="yes"

stop-word-threshold="2E6"

position-stop-word-threshold="2E7">

<authority weight="0" />

<freshness weight="0" />

<composite-rank composite-field-ref="content">

<proximity weight="50" />

<context weight="50">

<field-weight field-ref="fieldgroup1" value="20"/>

<field-weight field-ref="fieldgroup2" value="30"/>

<field-weight field-ref="fieldgroup3" value="50"/>

<field-weight field-ref="fieldgroup4" value="70"/>

<field-weight field-ref="fieldgroup5" value="20"/>

<field-weight field-ref="fieldgroup6" value="30"/>

</context>

</composite-rank>

</rank-profile>

## Navigators

Current implementation has 6 numeric navigators and 51 string type navigator. The objective of using navigators in the current implementation is not only filtering but a quick overview of number of documents that contains at least one instance of the filter element. Navigators that uses multi valued fields, it is ensured that values are de-duplicated so that count that shows on the filter item matches with search result count. List of all navigators are as follows.

<numeric-navigator name="amountpaidnavigator" display="Amount Paid" algorithm="manual">

<manual-cut value="10"/>

<manual-cut value="100"/>

<manual-cut value="1000"/>

<manual-cut value="10000"/>

<manual-cut value="20000"/>

<field-ref name="amountpaid"/>

</numeric-navigator>

<numeric-navigator name="extamountpaidnav" display="Extracted Amount Paid" algorithm="manual">

<manual-cut value="10"/>

<manual-cut value="100"/>

<manual-cut value="1000"/>

<manual-cut value="10000"/>

<manual-cut value="20000"/>

<field-ref name="extamountpaid"/>

</numeric-navigator>

<numeric-navigator name="amountinvolvednav" display="Amount Involved" algorithm="manual">

<manual-cut value="10"/>

<manual-cut value="100"/>

<manual-cut value="1000"/>

<manual-cut value="10000"/>

<manual-cut value="20000"/>

<field-ref name="amountinvolved"/>

</numeric-navigator>

<numeric-navigator name="yearquarternav" display="Qtr of Year" algorithm="manual">

<field-ref name="yearquarter"/>

</numeric-navigator>

<numeric-navigator name="createdyearnav" display="Created Year" algorithm="manual">

<field-ref name="createdyear"/>

</numeric-navigator>

<string-navigator name="consumeragerangenavigator" display="Consumer Age Range">

<field-ref name="consumeragerangenav"/>

</string-navigator>

<string-navigator name="consumercitynavigator" display="Consumer City">

<field-ref name="consumercitynav"/>

</string-navigator>

<string-navigator name="consumerzipnavigator" display="Consumer Zip">

<field-ref name="consumerzipnav"/>

</string-navigator>

<string-navigator name="subjectlocationnavigator" display="Subject Location" sort-by="frequency" sort-order="descending">

<field-ref name="subjectlocation"/>

</string-navigator>

<string-navigator name="subjectemailnavigator" display="Subject Email" sort-by="frequency" sort-order="descending">

<field-ref name="subjectemaildomain"/>

</string-navigator>

<string-navigator name="subjecturlnavigator" display="Subject URL" sort-by="frequency" sort-order="descending">

<field-ref name="subjecturl"/>

</string-navigator>

<string-navigator name="subjectcitynavigator" display="Subject City" sort-by="frequency" sort-order="descending">

<field-ref name="subjectcity"/>

</string-navigator>

<string-navigator name="subjectcountrynavigator" display="Subject Country" sort-by="frequency" sort-order="descending">

<field-ref name="subjectcountry"/>

</string-navigator>

<string-navigator name="subjectnamenavigator" display="Subject Name" sort-by="frequency" sort-order="descending">

<field-ref name="subjectname"/>

</string-navigator>

<string-navigator name="subjectstatenavigator" display="Subject State" sort-by="frequency" sort-order="descending">

<field-ref name="subjectstate"/>

</string-navigator>

<string-navigator name="subjectphonenavigator" display="Subject Phone" sort-by="frequency" sort-order="descending">

<field-ref name="subjectphonenav"/>

</string-navigator>

<string-navigator name="organizationnavigator" display="Complaint Source" sort-by="frequency" sort-order="descending">

<field-ref name="organization"/>

</string-navigator>

<string-navigator name="prodservicedescnavigator" display="Product Service" sort-by="frequency" sort-order="descending">

<field-ref name="prodservicedescnav"/>

</string-navigator>

<string-navigator name="statutenavigator" display="Statute" sort-by="frequency" sort-order="descending">

<field-ref name="statutenav"/>

</string-navigator>

<string-navigator name="topicsnavigator" display="Topic" sort-by="frequency" sort-order="descending">

<field-ref name="topicnav"/>

</string-navigator>

<string-navigator name="violationnavigator" display="Violation" sort-by="frequency" sort-order="descending">

<field-ref name="violationnav"/>

</string-navigator>

<string-navigator name="institutiontypenavigator" display="Institution Type" sort-by="frequency" sort-order="descending">

<field-ref name="institutiontypenav"/>

</string-navigator>

<string-navigator name="theftsubtypenavigator" display="Theft Subtype" sort-by="frequency" sort-order="descending">

<field-ref name="theftsubtype"/>

</string-navigator>

<string-navigator name="recordtypenavigator" display="Record Type" sort-by="frequency" sort-order="descending">

<field-ref name="recordtype"/>

</string-navigator>

<string-navigator name="associatedcompanycitynavigator" display="City" sort-by="frequency" sort-order="descending">

<field-ref name="associatedcompanycitynav"/>

</string-navigator>

<string-navigator name="associatedcompanycountrynavigator" display="Country" sort-by="frequency" sort-order="descending">

<field-ref name="associatedcompanycountrynav"/>

</string-navigator>

<string-navigator name="associatedcompanynamenavigator" display="Name" sort-by="frequency" sort-order="descending">

<field-ref name="associatedcompanynamenav"/>

</string-navigator>

<string-navigator name="associatedcompanybasenamenavigator" display="Name" sort-by="frequency" sort-order="descending">

<field-ref name="associatedcompanybasenamenav"/>

</string-navigator>

<string-navigator name="associatedcompanystatenavigator" display="State" sort-by="frequency" sort-order="descending">

<field-ref name="associatedcompanystatenav"/>

</string-navigator>

<string-navigator name="associatedcompanytypenavigator" display="Type" sort-by="frequency" sort-order="descending">

<field-ref name="associatedcompanytypenav"/>

</string-navigator>

<string-navigator name="institutionpscnavigator" display="Product Service Code" sort-by="frequency" sort-order="descending">

<field-ref name="institutionpscnav"/>

</string-navigator>

<string-navigator name="institutionvcnavigator" display="Violation Code" sort-by="frequency" sort-order="descending">

<field-ref name="institutionvcnav"/>

</string-navigator>

<string-navigator name="institutionscnavigator" display="Statute Code" sort-by="frequency" sort-order="descending">

<field-ref name="institutionscnav"/>

</string-navigator>

<string-navigator name="consumerstate" display="Consumer State" sort-by="frequency" sort-order="descending">

<field-ref name="consumerstatenav"/>

</string-navigator>

<string-navigator name="dncexistingbusinessrelnav" display="Done Business With" sort-by="frequency" sort-order="descending">

<field-ref name="dncexistingbusinessrel"/>

</string-navigator>

<string-navigator name="dncprerecmessagenav" display="Pre-recorded call?" sort-by="frequency" sort-order="descending">

<field-ref name="dncprerecmessage"/>

</string-navigator>

<string-navigator name="dncrequeststopcallingnav" display="Asked to stop calling" sort-by="frequency" sort-order="descending">

<field-ref name="dncrequeststopcalling"/>

</string-navigator>

<!-- Base name navigator -->

<string-navigator name="subjectbasenamenavigator" display="Subject Base Name" sort-by="frequency" sort-order="descending">

<field-ref name="subjectbasename"/>

</string-navigator>

<!-- Matched Base name navigator -->

<string-navigator name="matchedbasenamenavigator" display="Matched Base Name" sort-by="frequency" sort-order="descending">

<field-ref name="matchedbasename"/>

</string-navigator>

<!-- Matched Associated Company Base Name navigator -->

<string-navigator name="matchedassociatedbasenamenavigator" display="Matched Associated Company Base Name" sort-by="frequency" sort-order="descending">

<field-ref name="matchedassociatedbasename"/>

</string-navigator>

<!-- Entity extraction navigators. -->

<string-navigator name="companiesnavigator" display="Company">

<field-ref name="companies"/>

</string-navigator>

<string-navigator name="locationsnavigator" display="Location" >

<field-ref name="locations"/>

</string-navigator>

<string-navigator name="phrasesnavigator" display="Phrases" >

<field-ref name="extractedphrases"/>

</string-navigator>

<string-navigator name="entitiesnavigator" display="Entities" >

<field-ref name="extractedentities"/>

</string-navigator>

<string-navigator name="taxonomynavigator" display="Taxonomy" anchoring="prefix" sort-by="frequency" sort-order="descending">

<field-ref name="taxonomy"/>

</string-navigator>

<!-- Find Similar Complaints navigators -->

<string-navigator name="scopenavigatorsubjectaddress" display="Similar Address" deep="no" passive="yes" anchoring="none"

matcher="etc/resources/matching/scopenavigator.xml" use-quotes="always">

<field-ref name="pcaddressxml"/>

</string-navigator>

<string-navigator name="scopenavigatorreportedby" display="Similar Rep. Name" deep="no" passive="yes" anchoring="none"

matcher="etc/resources/matching/scopenavigator.xml" use-quotes="always">

<field-ref name="representativexml"/>

</string-navigator>

<!-- IDT Part II navigators -->

<string-navigator name="thefttypenavigator" display="Theft Type" sort-by="frequency" sort-order="descending">

<field-ref name="thefttypenav"/>

</string-navigator>

<string-navigator name="primarycompanyzipnavigator" display="Subject Zip" sort-by="frequency" sort-order="descending">

<field-ref name="primarycompanyzipnav"/>

</string-navigator>

<string-navigator name="extphraseentitynav" display="Extracted Phrases" sort-by="frequency" sort-order="descending">

<field-ref name="extphraseentity"/>

</string-navigator>

<string-navigator name="extnamedentitynav" display="Extracted Objects" sort-by="frequency" sort-order="descending">

<field-ref name="extnamedentity"/>

</string-navigator>

<string-navigator name="extcompaniesnav" display="Extracted Companies" sort-by="frequency" sort-order="descending">

<field-ref name="extcompanies"/>

</string-navigator>

<string-navigator name="extlocationsnav" display="Extracted Locations" sort-by="frequency" sort-order="descending">

<field-ref name="extlocations"/>

</string-navigator>

<string-navigator name="exturlsnav" display="Extracted Locations" sort-by="frequency" sort-order="descending">

<field-ref name="exturls"/>

</string-navigator>

<string-navigator name="extphrasesmultiwordnav" display="Extracted Phrases" sort-by="frequency" sort-order="descending">

<field-ref name="extphrasesmultiword"/>

</string-navigator>

<string-navigator name="extpaymentmethodnav" display="Extracted Phrases" sort-by="frequency" sort-order="descending">

<field-ref name="extpaymentmethod"/>

</string-navigator>

## Indexing

### CIS

For CIS indexing following pipeline is used. Details of individual processor stage has been provided in the appendix.

<pipeline name="CISMainPhonetic (webcluster)" default="0">

<priority>0</priority>

<processor name="DocInit"/>

<processor name="LMCOSystemIDAssigner"/>

<processor name="LMCOSystemIDToDocacl"/>

<processor name="ACLEncoder"/>

<processor name="LMCOLanguageAndEncodingDetector"/>

<processor name="LMCOInvalidCharactersChanger"/>

<processor name="LMCOInvalidCharactersChangerConXML"/>

<processor name="LMCOInvalidCharactersChangerACXML"/>

<processor name="ReadEntityDocuments"/>

<processor name="LMCOHTMLEncoder1"/>

<processor name="LMCOHTMLEncoder2"/>

<processor name="LMCOLTGTChanger"/>

<processor name="SentenceScopifier"/>

<processor name="AddSentenceNode"/>

<processor name="LMCOXMLParser"/>

<processor name="LMCOXMLScopifier"/>

<processor name="LMCOCopyCollectionName"/>

<processor name="LMCOCapitalizerMultivalue"/>

<processor name="LMCOMergeSubjectName"/>

<processor name="LMCOMergeSubjectCity"/>

<processor name="LMCOMergeSubjectState"/>

<processor name="LMCOMergeSubjectCountry"/>

<processor name="LMCOSubjectEmailMerger"/>

<processor name="LMCOSubjectURLMerger"/>

<processor name="LMCOConditionalCopy"/>

<processor name="LMAssociatedCompanyNameToNavCopy"/>

<processor name="LMCOAssociatedCompanyNameCopy"/>

<processor name="LMCOLowercaseMultivalue"/>

<processor name="LMCOURLRegExNormalizer1"/>

<processor name="LMCOURLRegExNormalizer2"/>

<processor name="LMCOURLRegExNormalizer2-1"/>

<processor name="LMCOURLRegExNormalizer3"/>

<processor name="LMCOURLRegExNormalizer4"/>

<processor name="LMCOURLRegExNormalizer5"/>

<processor name="LMCOSubjectPhoneNav"/>

<processor name="LMCOTildaRemoverSubectName"/>

<processor name="LMCOTildaRemoverSubjectCity"/>

<processor name="LMCOTildaRemoverSubjectState"/>

<processor name="LMCOTildaRemoverSubjectCountry"/>

<processor name="LMCORegexChangerSSN"/>

<processor name="LMCORegexChangerCC"/>

<processor name="FastHTMLParser"/>

<processor name="LMCOCompanyExtractor"/>

<processor name="LMCOCompanyPartialExtractor"/>

<processor name="LMCOCityExtractor"/>

<processor name="LMCOaddress2Remover1"/>

<processor name="LMCOaddress2Remover2"/>

<processor name="StreetSuffixRemover"/>

<processor name="LMCOSetProcessingTime"/>

<processor name="TeaserGenerator"/>

<processor name="Tokenizer(webcluster)"/>

<processor name="ScopeTokenizer(webcluster)"/>

<processor name="LMCOXMLifierCIS"/>

<processor name="Lemmatizer(webcluster)"/>

<processor name="LMCOSubjectNameNormalizerNew"/>

<processor name="LMCOAssociatedNameNormalizerNew"/>

<processor name="LMCOEntityNormalizer"/>

<processor name="LMCODuplicatesDeleter"/>

<processor name="LMCOSubjectNameStripper"/>

<processor name="LMCOCopyToSubjectBaseNameStrippedPhonetic"/>

<processor name="LMCOPhoneticNormalizer"/>

<processor name="LMCOPhoneticNormalizerSubjectCity"/>

<processor name="LMCOPhoneticNormalizerConsumerCity"/>

<processor name="CreateNgrams6"/>

<processor name="CreateNgrams8"/>

<processor name="CreateNgrams10"/>

<processor name="CreateNgrams12"/>

<processor name="CreateNgrams14"/>

<processor name="CreateNgramsSubjectName6"/>

<processor name="CreateNgramsSubjectName8"/>

<processor name="CreateNgramsSubjectName10"/>

<processor name="CreateNgramsSubjectName12"/>

<processor name="CreateNgramsSubjectName14"/>

<processor name="EncodingSubjectngramfield6"/>

<processor name="EncodingnormalizerSubjectngramfield8"/>

<processor name="EncodingnormalizerSubjectngramfield10"/>

<processor name="EncodingnormalizerSubjectngramfield12"/>

<processor name="EncodingnormalizerSubjectngramfield14"/>

<processor name="EntityVectorizer"/>

<processor name="Vectorizer(webcluster)"/>

<processor name="DateTimeNormalizer(webcluster)"/>

<processor name="DateTimeSelector(webcluster)"/>

<processor name="LMCODateTimeModifier"/>

<processor name="MapperTransformer"/>

<processor name="RankTuner"/>

<processor name="FIXMLGenerator"/>

<processor name="RTSOutput"/>

</pipeline>

### IDT Pipeline

<pipeline name="IDTMainScopeBaseNameFindSimilar (webcluster)" default="0">

<description><![CDATA[IDTMainScopeBaseName with Find Similar Stages]]></description>

<priority>0</priority>

<processor name="DocInit"/>

<processor name="LMCOSystemIDAssigner"/>

<processor name="LMCOSystemIDToDocacl"/>

<processor name="ACLEncoder"/>

<processor name="LMCOLTGTChanger"/>

<processor name="SentenceScopifier"/>

<processor name="AddSentenceNode"/>

<processor name="LMCOXMLParserIDT"/>

<processor name="LMCOXMLScopifierIDT"/>

<processor name="LMCOCopyCollectionName"/>

<processor name="LMCOCapitalizerMultivalue"/>

<processor name="LMCOMergeSuspectNameIDT"/>

<processor name="LMCOMergeSubjectCityIDT"/>

<processor name="LMCOMergeSubjectStateIDT"/>

<processor name="LMCOMergeSubjectCountryIDT"/>

<processor name="LMCOTildeRemoverSubjectPhone"/>

<processor name="LMCOSubjectPhoneNav"/>

<processor name="LMCOTildaRemoverSubjectState"/>

<processor name="LMCOTildaRemoverSubjectCountry"/>

<processor name="LMCOTildaRemoverSubjectCity"/>

<processor name="LMCOTildaRemoverSubectName"/>

<processor name="LMCOSpaceRemoverPCPhoneNo"/>

<processor name="LMCOSubjectEmailMerger"/>

<processor name="LMCOSubjectURLMerger"/>

<processor name="LMCOTildaRemoverSubjectEmailDomain"/>

<processor name="LMCOLowercaseMultivalue"/>

<processor name="LMCOURLRegExNormalizer1"/>

<processor name="LMCOURLRegExNormalizer2"/>

<processor name="LMCOURLRegExNormalizer2-1"/>

<processor name="LMCOURLRegExNormalizer3"/>

<processor name="LMCOURLRegExNormalizer4"/>

<processor name="LMCOURLRegExNormalizer5"/>

<processor name="LMCORegexChangerSSN"/>

<processor name="LMCORegexChangerCC"/>

<processor name="LMCOCompanyExtractor"/>

<processor name="LMCOCompanyPartialExtractor"/>

<processor name="LMCOCityExtractor"/>

<processor name="LMCOConditionalCopy"/>

<processor name="LMAssociatedCompanyNameToNavCopy"/>

<processor name="LMAssociatedCompanyNameNavCopy"/>

<processor name="LMCOaddress2Remover1"/>

<processor name="LMCOaddress2Remover2"/>

<processor name="StreetSuffixRemover"/>

<processor name="LMCOLanguageAndEncodingDetector"/>

<processor name="LMCOEncodingNormalizer"/>

<processor name="Lemmatizer(webcluster)"/>

<processor name="LMCOXMLifierIDT"/>

<processor name="LMCOSubjectNameNormalizerNew"/>

<processor name="LMCOAssociatedNameNormalizerNew"/>

<processor name="LMCOEntityNormalizer"/>

<processor name="LMCODuplicateDeleterIDT"/>

<processor name="LMCOTildaRemoverPrimaryCompanyZipNav"/>

<processor name="LMCOTildaRemoverAIPSC"/>

<processor name="LMCOTildaRemoverAISC"/>

<processor name="LMCOTildaRemoverAIVC"/>

<processor name="LMCODuplicateDeleterIDT2"/>

<processor name="LMCOSubjectNameStripper"/>

<processor name="LMCOCopyToSubjectBaseNameStrippedPhonetic"/>

<processor name="LMCOPhoneticNormalizer"/>

<processor name="CreateNgrams6"/>

<processor name="CreateNgrams8"/>

<processor name="CreateNgrams10"/>

<processor name="CreateNgrams12"/>

<processor name="CreateNgrams14"/>

<processor name="CreateNgramsSubjectName6"/>

<processor name="CreateNgramsSubjectName8"/>

<processor name="CreateNgramsSubjectName10"/>

<processor name="CreateNgramsSubjectName12"/>

<processor name="CreateNgramsSubjectName14"/>

<processor name="LMCOSetProcessingTime"/>

<processor name="TeaserGenerator"/>

<processor name="Tokenizer(webcluster)"/>

<processor name="ScopeTokenizer(webcluster)"/>

<processor name="EntityVectorizer"/>

<processor name="Vectorizer(webcluster)"/>

<processor name="DateTimeNormalizer(webcluster)"/>

<processor name="DateTimeSelector(webcluster)"/>

<processor name="LMCODateTimeModifier"/>

<processor name="MapperTransformer"/>

<processor name="RankTuner"/>

<processor name="FIXMLGenerator"/>

<processor name="RTSOutput"/>

</pipeline>

### DNC Pipeline

<pipeline name="DNCMainBaseNameFindSimilar (webcluster)" default="0">

<priority>0</priority>

<processor name="DocInit"/>

<processor name="LMCOSystemIDAssigner"/>

<processor name="LMCOSystemIDToDocacl"/>

<processor name="ACLEncoder"/>

<processor name="LMCOLTGTChanger"/>

<processor name="SentenceScopifier"/>

<processor name="AddSentenceNode"/>

<processor name="LMCOXMLParserDNC"/>

<processor name="LMCOXMLScopifierDNC"/>

<processor name="LMCOCopyCollectionName"/>

<processor name="LMCOCapitalizerMultivalueDNC"/>

<processor name="LMCORegexChangerSSN"/>

<processor name="LMCORegexChangerCC"/>

<processor name="LMCOCopyState"/>

<processor name="LMCOCopyName"/>

<processor name="LMCOConditionalCopy"/>

<processor name="LMCOEncodingNormalizer"/>

<processor name="LMCOMergeSubjectPhone"/>

<processor name="LMCOSubjectPhoneNav"/>

<processor name="LMCOSubjectEmailMerger"/>

<processor name="LMCOSubjectURLMerger"/>

<processor name="LMCOCompanyExtractor"/>

<processor name="LMCOCompanyPartialExtractor"/>

<processor name="LMCOCityExtractor"/>

<processor name="TeaserGenerator"/>

<processor name="Tokenizer(webcluster)"/>

<processor name="ScopeTokenizer(webcluster)"/>

<processor name="LMCOXMLifierDNC"/>

<processor name="Lemmatizer(webcluster)"/>

<processor name="LMCOSubjectNameNormalizerNew"/>

<processor name="LMCOEntityNormalizer"/>

<processor name="LMCODuplicatesDeleterDNC"/>

<processor name="LMCOSubjectNameStripper"/>

<processor name="LMCOCopyToSubjectBaseNameStrippedPhonetic"/>

<processor name="LMCOPhoneticNormalizer"/>

<processor name="CreateNgramsSubjectName6"/>

<processor name="CreateNgramsSubjectName7"/>

<processor name="CreateNgramsSubjectName8"/>

<processor name="CreateNgramsSubjectName9"/>

<processor name="CreateNgramsSubjectName10"/>

<processor name="CreateNgramsSubjectName11"/>

<processor name="CreateNgramsSubjectName12"/>

<processor name="CreateNgramsSubjectName13"/>

<processor name="CreateNgramsSubjectName14"/>

<processor name="LMCOSetProcessingTime"/>

<processor name="EntityVectorizer"/>

<processor name="Vectorizer(webcluster)"/>

<processor name="DateTimeNormalizer(webcluster)"/>

<processor name="DateTimeSelector(webcluster)"/>

<processor name="LMCODateTimeModifier"/>

<processor name="MapperTransformer"/>

<processor name="RankTuner"/>

<processor name="FIXMLGenerator"/>

<processor name="RTSOutput"/>

</pipeline>

## Processors

<processor name="**DocInit**" type="general" hidden="0">

<load module="processors.Basic" class="DocInit"/>

<config>

</config>

<description><![CDATA[Initialize document attributes. Does not overwrite existing values, except processingtime. The values are:

processingtime = int(time.time())

if data exists:

size = len(data)

]]></description>

<inputs>

</inputs>

</processor>

========================================================================================================

<processor name="**LMCOSystemIDAssigner**" type="general" hidden="0">

<load module="processors.Basic" class="AttributeAssigner"/>

<config>

<param name="Attribute" value="docaclsystemid" type="str"/>

<param name="Value" value="pri" type="str"/>

</config>

<description><![CDATA[Assign a constant value to a document attribute

The attribute name and value are defined in configuration parameters.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOSystemIDToDocacl**" type="general" hidden="0">

<load module="processors.Basic" class="Programmable"/>

<config>

<param name="Init" value="" type="str"/>

<param name="Program" value="doc.Set('docacl', &quot;pri&quot; + doc.GetValue('docacl').replace(&quot; &quot;,&quot; pri&quot;))" type="str"/>

</config>

<description><![CDATA[Execute any Python code

Execute the Python code in 'Program' where the variable 'doc' is set to the document. The context of that code can be initialized withe the 'Init' code.]]></description>

<inputs>

</inputs>

</processor>

========================================================================================================

<processor name="**ACLEncoder**" type="general" hidden="0">

<load module="processors.ACLEncoder" class="ACLEncoder"/>

<config>

<param name="DisableAttr" value="docaclisencoded" type="str"/>

<param name="EncodeUpperCase" value="0" type="int"/>

<param name="SecurityAttr" value="docacl" type="str"/>

<param name="SpacesPermitted" value="0" type="int"/>

</config>

<description><![CDATA[Encodes entries in document ACLs. The ACL Encoder processor is used (instead of the Document Security document processor) when connectors retrieve the Access Control List (ACL) for the documents and (some of) the connectors don't apply the encoding. The encoding is necessary because each entry (user or group name) must be a single, valid token, which may not contain non-alphabetic, non-numeric characters.

If a connector itself does produce encoded ACLs, it must set the document attribute identified by DisableAttr to true. This causes the ACL Encoder to leave the ACL in this document unchanged. Space characters are normally used as delimiters in the ACL string. If it is necessary to handle ACL entries with spaces, SpacesPermitted can be set to true. This will cause the ACL encoder to separate tokens based on the system identifier prefix. This will only work if all ACL entries are known to have the same system identifier, and is only intended as a workaround for existing connectors that are unable to encode names.

Parameters:

SecurityAttr - Document attribute used to store ACL.

DisableAttr - Document attribute used to signify that ACL already is encoded.

SpacesPermitted - If true, spaces in user/group names are permitted.

EncodeUppperCase - If true, group names with upper or mixed case are encoded, such that case-sensitive matching is enabled.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOLanguageAndEncodingDetector**" type="general" hidden="0">

<load module="processors.HTML" class="LanguageAndEncodingDetector"/>

<config>

<param name="ContentAttribute" value="subjectaddress" type="str"/>

<param name="FallbackEncoding" value="iso-8859-1" type="str"/>

<param name="FallbackLanguage" value="unknown" type="str"/>

<param name="URLAttribute" value="url" type="str"/>

</config>

<description><![CDATA[Automatic language and encoding detection of HTML, plain text or XML in field defined in ContentAttribute

The order of preference is: explicit metadata, content, and URL. If any

of the language or charset attributes are preset, the values are used to sanitize

the detection.

The fields in ContentAttribute are a preference list; only the first field that has content is processed

The FallbackEncoding and FallbackLanguage values are used if none of the above strategies detects an encoding or language, respectively.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOInvalidCharactersChanger**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value=" " type="str"/>

<param name="inputField" value="pcxml" type="str"/>

<param name="outputField" value="pcxml" type="str"/>

<param name="regex" value="&amp;#x\d+;" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the regex from the input field to "changeTo" and put resulting text into output field (usually both are &apos;data&apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOInvalidCharacterChangerconxml**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="" type="str"/>

<param name="inputField" value="conxml" type="str"/>

<param name="outputField" value="conxml" type="str"/>

<param name="regex" value="&amp;#x\d+;" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the regex from the input field to "changeTo" and put resulting text into output field (usually both are &apos;data&apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOInvalidCharactersChangerACXML**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value=" " type="str"/>

<param name="inputField" value="acxml" type="str"/>

<param name="outputField" value="acxml" type="str"/>

<param name="regex" value="&amp;#x\d+;" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the regex from the input field to "changeTo" and put resulting text into output field (usually both are &apos;data&apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOInvalidCharactersChangerConXML**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value=" " type="str"/>

<param name="inputField" value="conxml" type="str"/>

<param name="outputField" value="conxml" type="str"/>

<param name="regex" value="&amp;#x\d+;" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the regex from the input field to "changeTo" and put resulting text into output field (usually both are &apos;data&apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**ReadEntityDocuments**" type="general" hidden="0">

<load module="processors.ReadEntityDocuments" class="ReadEntityDocuments"/>

<config>

<param name="DocIdFieldName" value="contentid" type="str"/>

<param name="SourceDir" value="path//to//shared//location" type="str"/>

</config>

<description><![CDATA[ReadEntityDocuments

Reads JSON documents with extracted phrases and entities from shared location and populate corresponding FAST fields with extracted entities

Following fields are populated by this processor

- extractedphrases

- extcompanies

- extamountpaid

- extlocations

- exturls

Documents are identified by the docid in the shared location. For every record there would be one document with entities JSON.

If successfully extracted the document shall be deleted from the shared location. Otherwise the document shall be left there for processing in next iteration.

]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOHTMLEncoder1**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="&amp;lt;" type="str"/>

<param name="inputField" value="comments" type="str"/>

<param name="outputField" value="comments" type="str"/>

<param name="regex" value="&lt;" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the regex from the input field to "changeTo" and put resulting text into output field (usually both are &apos;data&apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOHTMLEncoder2**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="&amp;gt;" type="str"/>

<param name="inputField" value="comments" type="str"/>

<param name="outputField" value="comments" type="str"/>

<param name="regex" value="&gt;" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the regex from the input field to "changeTo" and put resulting text into output field (usually both are &apos;data&apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOLTGTChanger**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value=" " type="str"/>

<param name="inputField" value="comments" type="str"/>

<param name="outputField" value="commentxml" type="str"/>

<param name="regex" value="&lt;|&gt;" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the regex from the input field to &amp;quot;changeTo&amp;quot; and put resulting text into output field (usually both are &amp;amp;apos;data&amp;amp;apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**SentenceScopifier**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="&lt;/sentence&gt;&lt;sentence&gt;" type="str"/>

<param name="inputField" value="commentxml" type="str"/>

<param name="outputField" value="commentxml" type="str"/>

<param name="regex" value="(\.|\?|\!)\s|[\r\n]" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the regex from the input field to &amp;amp;quot;changeTo&amp;amp;quot; and put resulting text into output field (usually both are &amp;amp;amp;apos;data&amp;amp;amp;apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**AddSentenceNode**" type="general" hidden="0">

<load module="processors.AddPrefixSuffix" class="AddPrefixSuffix"/>

<config>

<param name="AttributeName" value="commentxml" type="str"/>

<param name="Prefix" value="&lt;comment&gt;&lt;sentence&gt;" type="str"/>

<param name="Suffix" value="&lt;/sentence&gt;&lt;/comment&gt;" type="str"/>

</config>

<description><![CDATA[Add static prefix and suffix text to attribute value. Pankaj Bose]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOXMLParser**" type="general" hidden="0">

<load module="processors.XMLDP" class="XMLParser"/>

<config>

<param name="Attributes" value="conxml:conxml pcxml:pcxml acxml:acxml representativexml:representativexml pcaddressxml:pcaddressxml" type="str"/>

<param name="MaxErrors" value="20" type="int"/>

<param name="Strict" value="1" type="int"/>

</config>

<description><![CDATA[Parses XML and creates a DOM tree

The Attributes mapping is a sequence of mappings src[:dst] where the destination is optional. With only a source, the DOM tree is added in the &amp;amp;amp;amp;amp;quot;dom&amp;amp;amp;amp;amp;quot; metadata value of the source attribute. With a destination attribute, the attribute gets the DOM tree value.

The Strict flag determines whether any XML error shall create a document error.

The first MaxErrors parse error messages are put in the document log.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOXMLScopifier**" type="general" hidden="0">

<load module="processors.XMLDP" class="XMLScopifier"/>

<config>

<param name="AttributeProperties" value="conxml:minute pcxml:minute acxml:minute pcaddressxml:minute representativexml:minute" type="str"/>

<param name="AutoRealType" value="float" type="str"/>

<param name="AutoTyping" value="0" type="int"/>

<param name="Mapping" value="conxml:conxml pcxml:pcxml acxml:acxml pcaddressxml:pcaddressxml representativexml:representativexml" type="str"/>

<param name="PreserveWhitespace" value="0" type="int"/>

<param name="Separator" value="0" type="int"/>

<param name="TypeAttribute" value="" type="str"/>

<param name="TypeMap" value="" type="str"/>

</config>

<description><![CDATA[Automatically configured for cluster]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOCopyCollectionName**" type="general" hidden="0">

<load module="processors.CopyCollectionName" class="CopyCollectionName"/>

<config>

<param name="outputField" value="recordtype" type="str"/>

</config>

<description><![CDATA[Copies the collection name to output parameter]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOCapitalizerMultivalue**" type="general" hidden="0">

<load module="processors.CapitalizerMultivalue" class="CapitalizerMultivalue"/>

<config>

<param name="Input" value="primarycompanyname primarycompanycity primarycompanystate primarycompanycountry associatedcompanyname associatedcompanycity associatedcompanystate associatedcompanycountry" type="str"/>

<param name="Output" value="primarycompanyname primarycompanycity primarycompanystate primarycompanycountry associatedcompanyname associatedcompanycity associatedcompanystate associatedcompanycountry" type="str"/>

<param name="Separator" value="|" type="str"/>

</config>

<description><![CDATA[Capitalize values in multivalue fields

Capitalize values specified in the Input field.

and stores capitalized value into Output field

Examples: LONG BEACH||BOSTON|new york --> Long Beach||Boston|New York]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOMergeSubjectName**" type="general" hidden="0">

<load module="processors.AttributeMergerMultivalue" class="AttributeMergerMultivalue"/>

<config>

<param name="Input" value="primarycompanyname associatedcompanyname" type="str"/>

<param name="Output" value="subjectname" type="str"/>

<param name="Separator" value="|" type="str"/>

</config>

<description><![CDATA[Merge Several Document Multivalue Attributes

The document attributes named in the Input parameter, in the form of "a b c", are merged together and placed in the Output attribute.

The values of a, b, and c are merged together in the order specified and are seperated by the Separator parameter.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOMergeSubjectCity**" type="general" hidden="0">

<load module="processors.AttributeMergerMultivalue" class="AttributeMergerMultivalue"/>

<config>

<param name="Input" value="primarycompanycity associatedcompanycity" type="str"/>

<param name="Output" value="subjectcity" type="str"/>

<param name="Separator" value="|" type="str"/>

</config>

<description><![CDATA[Merge Several Document Multivalue Attributes

The document attributes named in the Input parameter, in the form of "a b c", are merged together and placed in the Output attribute.

The values of a, b, and c are merged together in the order specified and are seperated by the Separator parameter.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOMergeSubjectState**" type="general" hidden="0">

<load module="processors.AttributeMergerMultivalue" class="AttributeMergerMultivalue"/>

<config>

<param name="Input" value="primarycompanystate associatedcompanystate" type="str"/>

<param name="Output" value="subjectstate" type="str"/>

<param name="Separator" value="|" type="str"/>

</config>

<description><![CDATA[Merge Several Document Multivalue Attributes

The document attributes named in the Input parameter, in the form of "a b c", are merged together and placed in the Output attribute.

The values of a, b, and c are merged together in the order specified and are seperated by the Separator parameter.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOMergeSubjectCountry**" type="general" hidden="0">

<load module="processors.AttributeMergerMultivalue" class="AttributeMergerMultivalue"/>

<config>

<param name="Input" value="primarycompanycountry associatedcompanycountry" type="str"/>

<param name="Output" value="subjectcountry" type="str"/>

<param name="Separator" value="|" type="str"/>

</config>

<description><![CDATA[Merge Several Document Multivalue Attributes

The document attributes named in the Input parameter, in the form of "a b c", are merged together and placed in the Output attribute.

The values of a, b, and c are merged together in the order specified and are seperated by the Separator parameter.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOSubjectEmailMerger**" type="general" hidden="0">

<load module="processors.AttributeMergerMultivalue" class="AttributeMergerMultivalue"/>

<config>

<param name="Input" value="primarycompanyemail associatedcompanyemail" type="str"/>

<param name="Output" value="subjectemaildomain" type="str"/>

<param name="Separator" value="|" type="str"/>

</config>

<description><![CDATA[Merge Several Document Multivalue Attributes

The document attributes named in the Input parameter, in the form of &quot;a b c&quot;, are merged together and placed in the Output attribute.

The values of a, b, and c are merged together in the order specified and are seperated by the Separator parameter.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOSubjectURLMerger**" type="general" hidden="0">

<load module="processors.AttributeMergerMultivalue" class="AttributeMergerMultivalue"/>

<config>

<param name="Input" value="primarycompanyurl associatedcompanyurl" type="str"/>

<param name="Output" value="subjecturl" type="str"/>

<param name="Separator" value="|" type="str"/>

</config>

<description><![CDATA[Merge Several Document Multivalue Attributes

The document attributes named in the Input parameter, in the form of &quot;a b c&quot;, are merged together and placed in the Output attribute.

The values of a, b, and c are merged together in the order specified and are seperated by the Separator parameter.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOConditionalCopy**" type="general" hidden="0">

<load module="processors.ConditionalCopy" class="ConditionalCopy"/>

<config>

<param name="fromField" value="subjectname" type="str"/>

<param name="toField" value="subjectbasename" type="str"/>

</config>

<description><![CDATA[Copy the value from the From field to To field (usually both are &apos;data&apos;) where To Field doesn't have already set value. Pankaj Bose]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMAssociatedCompanyNameToNavCopy**" type="general" hidden="0">

<load module="processors.Basic" class="AttributeCopy"/>

<config>

<param name="Attributes" value="" type="str"/>

<param name="Input" value="associatedcompanyname" type="str"/>

<param name="Output" value="associatedcompanynamenav" type="str"/>

</config>

<description><![CDATA[Copy document attributes

The document attribute named by the Input configuration parameter

is copied to the attribute named by the Output parameter. Any existing

values in the output is lost. If the input attribute does not exist,

the operation is ignored

Several attributes can be copied using the Attributes parameter. It

has the form &quot;a:b c:d&quot;, where a is copied to b and c is copied to d. Any

number of such mappings can be declared. The Attributes copy is performed

first, then the Input to Output copy.

From attributes starting with a &#039;\*&#039; are looked up in the routing attributes (excluding the leading &#039;\*&#039;), allowing routing attributes to be copied to normal document attributes.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOAssociatedCompanyNameCopy**" type="general" hidden="0">

<load module="processors.Basic" class="AttributeCopy"/>

<config>

<param name="Attributes" value="" type="str"/>

<param name="Input" value="associatedcompanyname" type="str"/>

<param name="Output" value="associatedcompanybasenamenav" type="str"/>

</config>

<description><![CDATA[Copy document attributes

The document attribute named by the Input configuration parameter

is copied to the attribute named by the Output parameter. Any existing

values in the output is lost. If the input attribute does not exist,

the operation is ignored

Several attributes can be copied using the Attributes parameter. It

has the form &quot;a:b c:d&quot;, where a is copied to b and c is copied to d. Any

number of such mappings can be declared. The Attributes copy is performed

first, then the Input to Output copy.

From attributes starting with a &#039;\*&#039; are looked up in the routing attributes (excluding the leading &#039;\*&#039;), allowing routing attributes to be copied to normal document attributes.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOLowercaseMultivalue**" type="general" hidden="0">

<load module="processors.LowercaseMultivalue" class="LowercaseMultivalue"/>

<config>

<param name="Input" value="subjectemaildomain subjecturl" type="str"/>

<param name="Output" value="subjectemaildomain subjecturl" type="str"/>

<param name="Separator" value="|" type="str"/>

</config>

<description><![CDATA[Lowercase values in multivalue fields

Lowercase values specified in the Input field.

and stores lowercased value into Output field

Examples: LONG BEACH||BOSTON|new york --&gt; long beach||boston|new york]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOURLRegExNormalizer1**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="|" type="str"/>

<param name="inputField" value="subjecturl" type="str"/>

<param name="outputField" value="subjecturl" type="str"/>

<param name="regex" value="/\|" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the regex from the input field to &quot;changeTo&quot; and put resulting text into output field (usually both are &amp;apos;data&amp;apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOURLRegExNormalizer2**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="" type="str"/>

<param name="inputField" value="subjecturl" type="str"/>

<param name="outputField" value="subjecturl" type="str"/>

<param name="regex" value="((http:|https:)//|/$)" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the regex from the input field to &amp;amp;quot;changeTo&amp;amp;quot; and put resulting text into output field (usually both are &amp;amp;amp;apos;data&amp;amp;amp;apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOURLRegExNormalizer2**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="" type="str"/>

<param name="inputField" value="subjecturl" type="str"/>

<param name="outputField" value="subjecturl" type="str"/>

<param name="regex" value="((http:|https:)//|/$)" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the regex from the input field to &amp;amp;quot;changeTo&amp;amp;quot; and put resulting text into output field (usually both are &amp;amp;amp;apos;data&amp;amp;amp;apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOURLRegExNormalizer3**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="|" type="str"/>

<param name="inputField" value="subjecturl" type="str"/>

<param name="outputField" value="subjecturl" type="str"/>

<param name="regex" value="(\.[^.]+$)|(\.[^.]+\|)" type="str"/>

</config>

<description><![CDATA[Replace domain (.com etc) or .com etc follwoed by | with |]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOURLRegExNormalizer4**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="" type="str"/>

<param name="inputField" value="subjecturl" type="str"/>

<param name="outputField" value="subjecturl" type="str"/>

<param name="regex" value="\|$" type="str"/>

</config>

<description><![CDATA[Replace | from the end of the string]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOURLRegExNormalizer5**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="" type="str"/>

<param name="inputField" value="subjecturl" type="str"/>

<param name="outputField" value="subjecturl" type="str"/>

<param name="regex" value="(www\.)" type="str"/>

</config>

<description><![CDATA[Replace www from the end of the string]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOSubjectPhoneNav**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="" type="str"/>

<param name="inputField" value="subjectphone" type="str"/>

<param name="outputField" value="subjectphonenav" type="str"/>

<param name="regex" value="\s+|-" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the regex from the input field to "changeTo" and put resulting text into output field (usually both are &apos;data&apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOTildaRemoverSubectName**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="|" type="str"/>

<param name="inputField" value="subjectname" type="str"/>

<param name="outputField" value="subjectname" type="str"/>

<param name="regex" value="~" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the regex from the input field to "changeTo" and put resulting text into output field (usually both are &apos;data&apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOTildaRemoverSubjectCity**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="|" type="str"/>

<param name="inputField" value="subjectcity" type="str"/>

<param name="outputField" value="subjectcity" type="str"/>

<param name="regex" value="~" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the regex from the input field to "changeTo" and put resulting text into output field (usually both are &apos;data&apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOTildaRemoverSubjectState**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="|" type="str"/>

<param name="inputField" value="subjectstate" type="str"/>

<param name="outputField" value="subjectstate" type="str"/>

<param name="regex" value="~" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the regex from the input field to "changeTo" and put resulting text into output field (usually both are &apos;data&apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOTildaRemoverSubjectCountry**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="|" type="str"/>

<param name="inputField" value="subjectcountry" type="str"/>

<param name="outputField" value="subjectcountry" type="str"/>

<param name="regex" value="~" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the regex from the input field to "changeTo" and put resulting text into output field (usually both are &apos;data&apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="LMCORegexChangerSSN" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="XXX-XX-XXXX" type="str"/>

<param name="inputField" value="comments" type="str"/>

<param name="outputField" value="redactedcomments" type="str"/>

<param name="regex" value="[0-9]{3}[-\s][0-9]{2}[-\s][0-9]{4}" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the regex from the input field to "changeTo" and put resulting text into output field (usually both are &apos;data&apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCORegexChangerCC**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="xxxx-xxxx-xxxx-xxxx" type="str"/>

<param name="inputField" value="redactedcomments" type="str"/>

<param name="outputField" value="redactedcomments" type="str"/>

<param name="regex" value="\b4[0-9]{12}(?:[0-9]{3})?\b|\b4\d{3}[ .#-]?\d{4}[ .#-]?\d{4}[ .#-]?\d{4}\b|\b5[1-5]\d{2}[ .#-]?\d{4}[ .#-]?\d{4}[ .#-]?\d{4}\b|\b6011[ .#-]?\d{4}[ .#-]?\d{4}[ .#-]?\d{4}\b|\b3[0,6,8]\d{12}\b|\b3[4,7]\d{13}\b|\b(?:2131|1800|35\d{3})\d{11}\b" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the regex from the input field to "changeTo" and put resulting text into output field (usually both are &apos;data&apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**FastHTMLParser**" type="general" hidden="0">

<load module="processors.FastHTMLParser" class="FastHTMLParser"/>

<config>

<param name="HandleBRTag" value="" type="str"/>

<param name="Handletags" value="" type="str"/>

<param name="Input" value="html" type="str"/>

<param name="MaxChunks" value="10000" type="int"/>

<param name="MaxJoinChunkLen" value="1000" type="int"/>

<param name="MaxTitleLen" value="100" type="int"/>

<param name="MergeTableRows" value="off" type="str"/>

<param name="URL" value="url" type="str"/>

</config>

<description><![CDATA[Parse the HTML and extract title, body, headings, links, and metadata.

Extracts document attributes from HTML content.

After processing, the document will contain the fields

'title', 'headings', and 'body'.

It will also contain one field per meta tag extracted

from the input as well as a 'links' field containing a

a TextChunks object with anchor text blocks as chunks

and the corresponding href stored in their meta dictionary.

If 'HandleBRTag' is set to 'sentence', the parser will not

consider BR tags to be paragraph-separating (unless there are

two or more consecutive). The 'Handletags' parameter (if set)

should contain a string representation of a python dictionary.

A key in the dictionary specifies a tag, and the corresponding

string value what the parser should output to the body field

when such a tag is encountered. E.g {'i': ' '} would cause

a single space to be printed when a <i> tag is encountered.

Please note that you might override the parser's handling (though

not everything can be overrided) of the tags, so use with caution.

'MergeTableRows' controls whether cells in a table row

should be merged together into a single paragraph or remain as

individual paragraphs. Possible values are 'off' and 'on'.

NOTE: If a cell contains paragraph breaking markup and not just

text (e.g. a <div> or <p> tag), that markup will still

cause a paragraph break between the cells even if merging is enabled.

'MaxChunks' sets an upper bound for the number of chunks a document might

contain before chunks are joined to larger chunks

'MaxJoinChunkLen' is the maximum size of a chunk that can be joined with another chunk.

Larger chunks will not be joined with others.

]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOCompanyExtractor**" type="general" hidden="0">

<load module="processors.linguistics.Matching" class="Matcher"/>

<config>

<param name="byteguard" value="" type="str"/>

<param name="dispatch" value="language" type="str"/>

<param name="filter" value="airline:intersects company:intersects person:intersects \*:illegal" type="str"/>

<param name="guard" value="comments" type="str"/>

<param name="input" value="comments" type="str"/>

<param name="lazy" value="0" type="int"/>

<param name="matcher" value="en:linguistics/extractors/configuration.companyextractor.pass1.en.xml de:linguistics/extractors/configuration.companyextractor.pass1.de.xml es:linguistics/extractors/configuration.companyextractor.pass1.es.xml fr:linguistics/extractors/configuration.companyextractor.pass1.fr.xml it:linguistics/extractors/configuration.companyextractor.pass1.it.xml nl:linguistics/extractors/configuration.companyextractor.pass1.nl.xml pt:linguistics/extractors/configuration.companyextractor.pass1.pt.xml \*:linguistics/extractors/configuration.companyextractor.pass1.en.xml" type="str"/>

<param name="meta" value="type:semantic" type="str"/>

<param name="output" value="companies" type="str"/>

<param name="phrases" value="0" type="str"/>

<param name="rename" value="meta:ticker" type="str"/>

<param name="separator" value="|" type="str"/>

<param name="separator2" value="/" type="str"/>

<param name="type" value="company" type="str"/>

</config>

<description><![CDATA[Extracts names of companies, based on general patterns and dictionaries.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOCompanyPartialExtractor**" type="general" hidden="0">

<load module="processors.linguistics.Matching" class="PartialMatcher"/>

<config>

<param name="base" value="base" type="str"/>

<param name="filter" value="company:intersects \*:illegal" type="str"/>

<param name="inherit" value="0" type="int"/>

<param name="input" value="comments" type="str"/>

<param name="meta" value="type:semantic" type="str"/>

<param name="mode" value="strict" type="str"/>

<param name="partial" value="partial" type="str"/>

<param name="regexlimit" value="" type="str"/>

<param name="rejector" value="" type="str"/>

<param name="type" value="company" type="str"/>

</config>

<description><![CDATA[Extracts names of companies, based on resolving partial forms.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOCityExtractor**" type="general" hidden="0">

<load module="processors.linguistics.Matching" class="Matcher"/>

<config>

<param name="byteguard" value="" type="str"/>

<param name="dispatch" value="language" type="str"/>

<param name="filter" value="" type="str"/>

<param name="guard" value="comments" type="str"/>

<param name="input" value="comments" type="str"/>

<param name="lazy" value="0" type="int"/>

<param name="matcher" value="en:linguistics/extractors/configuration.locationextractor.pass1.en.xml de:linguistics/extractors/configuration.locationextractor.pass1.de.xml es:linguistics/extractors/configuration.locationextractor.pass1.es.xml fr:linguistics/extractors/configuration.locationextractor.pass1.fr.xml it:linguistics/extractors/configuration.locationextractor.pass1.it.xml nl:linguistics/extractors/configuration.locationextractor.pass1.nl.xml pt:linguistics/extractors/configuration.locationextractor.pass1.pt.xml \*:linguistics/extractors/configuration.locationextractor.pass1.indep.xml" type="str"/>

<param name="meta" value="type:semantic" type="str"/>

<param name="output" value="locations" type="str"/>

<param name="phrases" value="0" type="str"/>

<param name="rename" value="a:country b:region c:subregion d:class g:coordinates" type="str"/>

<param name="separator" value="|" type="str"/>

<param name="separator2" value="/" type="str"/>

<param name="type" value="location" type="str"/>

</config>

<description><![CDATA[Extracts names of geographical locations, based on general patterns and dictionaries.

LMCO NOTES:

- No filter values since we are not extracting from an attribute from which companies or another entity has extracted entities.

- Input is companycity, we are focusing on this field to extract valid city names.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOaddress2Remover1**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="" type="str"/>

<param name="inputField" value="subjectaddress" type="str"/>

<param name="outputField" value="subjectaddress" type="str"/>

<param name="regex" value="(suites|suite|apt|apt#|apt.|unit|flat|room|block)(\s|#)([a-zA-Z0-9\_\-]{1,32})" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the suite, floor, unit, Apt, etc from the input field to &amp;quot;changeTo&amp;quot; and put resulting text into output field (usually both are &amp;amp;apos;data&amp;amp;apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOaddress2Remover2**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="" type="str"/>

<param name="inputField" value="subjectaddress" type="str"/>

<param name="outputField" value="subjectaddress" type="str"/>

<param name="regex" value="([a-zA-Z0-9\_\-]{1,32})\s(floor|floors|flr#|flr.)" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the from the input field to &amp;quot;changeTo&amp;quot; and put resulting text into output field (usually both are &amp;amp;apos;data&amp;amp;apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**StreetSuffixRemover**" type="general" hidden="0">

<load module="processors.StopWordsRemover" class="StopWordsRemover"/>

<config>

<param name="Attribute" value="subjectaddress" type="str"/>

<param name="Separator" value="|" type="str"/>

<param name="StopwordFilePath" value="D:\esp\resources\entitynormalizer\StreetSuffixLists.txt" type="str"/>

</config>

<description><![CDATA[StopWordsRemover

Removes stop words from an entity. The stopwords are saved in a stopword file.

The file path need to be provided for the parameter &quot;StopWordFilePath&quot;

The Attribute parameter takes the name of the fields. It can take multiple field name separated by

space.

The Separator field takes the separator chracter for the multivalue field.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOSetProcessingTime**" type="general" hidden="0">

<load module="processors.Basic" class="Programmable"/>

<config>

<param name="Init" value="" type="str"/>

<param name="Program" value="doc.Set('processingtime', time.strftime(&quot;%Y-%m-%d %H:%M:%S&quot;))" type="str"/>

</config>

<description><![CDATA[Execute any Python code

Execute the Python code in 'Program' where the variable 'doc' is set to the document. The context of that code can be initialized withe the 'Init' code.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**TeaserGenerator**" type="general" hidden="0">

<load module="processors.HTML" class="TeaserGenerator"/>

<config>

<param name="MaxLen" value="300" type="int"/>

<param name="MinLen" value="75" type="int"/>

</config>

<description><![CDATA[Teaser generator summarizing the document

The strategy is to first use the description metadata. If that does

not exist or is too small, then automatically extract a teaser from the

body avoiding text that typically appears in navigation menus. Again, if

that fails or is too small, use the first words in the body. It does not

overwrite an existing teaser attribute in the document.

Configuration parameters:

MaxLen: The teaser will be chopped at a word boundary at or below MaxLen.

MinLen: Minimum length of teaser. If the length of the teaser from a phase

above is less than MinLen, the next strategy is applied.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**Tokenizer**" type="general" hidden="0">

<load module="processors.linguistics.Tokenization" class="Tokenizer"/>

<config>

<param name="compactthreshold" value="1048576" type="int"/>

<param name="dynamic" value="body" type="str"/>

<param name="preferred\_languages" value="ja ko zh-simplified zh-traditional th" type="str"/>

<param name="tokenize" value="title:title headings:headings body:body" type="str"/>

<param name="vectorize" value="" type="str"/>

</config>

<description><![CDATA[Tokenizes (and optionally produces additional data for the tokens) attributes.

The parameter 'tokenize' indicates which document fields to segment

into tokens. It has either the form a:b or a:b:c. Where a is the document

field to be tokenized and b is the name of the field in the index. Though

it is possible to give different names to document fields during indexing,

usually a and b will be equal since document fields and index fields

have the same name most of the time. In the second form, a:b:c the c is the

tokenization mode to apply: this mode has to be set-up correctly in

$FASTSEARCH/etc/tokenizer/tokenization.xml.

The parameter 'dynamic' indicates which document fields will be used

for dynamic hit highlighting later. Special tokenization has to be applied

to those.

The parameter 'compacttreshold' specifies the minimum document size for which

documents are processed in a memory saving mode. This memory saving mode is slower,

but allows feeding of large documents.

The parameter 'preferred\_languages' controls behaviour of the tokenizer for

multilingual documents. In case a multilingual document has any of the languages

listed in 'preferred\_languages' as secondary language, tokenization will be

based on that language, instead of the primary language. Enables correct CJK processing

for multilingual documents.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**ScopeTokenizer**" type="general" hidden="0">

<load module="processors.linguistics.Tokenization" class="ScopeTokenizer"/>

<config>

<param name="compactthreshold" value="1048576" type="int"/>

<param name="preferred\_languages" value="ja ko zh-simplified zh-traditional th" type="str"/>

<param name="tokenize" value="xml" type="str"/>

</config>

<description><![CDATA[Tokenization of text scopes

The parameter 'tokenize' indicates which document fields to segment

into tokens. These fields have to be listed space-separated. It is possible

to specify an alternative tokenization mode for each field by appending

its id to this field-name in the list, like a:b. Where a is the document

field to be tokenized and b is the tokenization mode to apply: this mode

has to be set-up correctly in $FASTSEARCH/etc/tokenizer/tokenization.xml.

The parameter 'preferred\_languages' controls behaviour of the tokenizer for

multilingual documents. In case a multilingual document has any of the languages

listed in 'preferred\_languages' as secondary language, tokenization will be

based on that language, instead of the primary language. Enables correct CJK processing

for multilingual documents.

The parameter 'compacttreshold' specifies the minimum document size for which

documents are processed in a memory saving mode. This memory saving mode is slower,

but allows feeding of large documents.

The configuration of the tokenizers requires advanced knowledge of FAST

ESP.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOXMLifierCIS**" type="general" hidden="0">

<load module="processors.linguistics.Scopification" class="XMLifier"/>

<config>

<param name="limit" value="1024" type="int"/>

<param name="mappings" value="conxml:resconxml pcxml:respcxml acxml:resacxml representativexml:resrepresentativexml pcaddressxml:respcaddressxml" type="str"/>

<param name="metaprefix" value="@" type="str"/>

</config>

<description><![CDATA[Creates an XML rendering of a specified scope type field.

Given a scope type field as produced by the Scopifier processor, creates a string type field containing an XML rendering of the scope field.

An upper limit of the rendered XML string can be given, specified in kB. If the rendered XML is longer than this, the output field instead gets

populated with a small XML error message. The purpose of this is to avoid illegal XML in a document summary field, if the size of the document summary

field is insufficient to store the full XML string. The value of the &amp;#039;limit&amp;#039; parameter should be synchronized with the maximum document summary field

length specified in the index profile.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**Lemmatizer**" type="general" hidden="0">

<load module="processors.linguistics.Lemmatization" class="Lemmatizer"/>

<config>

<param name="attributes" value="title headings body" type="str"/>

<param name="configfile" value="etc/LemmatizationConfig.xml" type="str"/>

<param name="fds4compatibility" value="0" type="int"/>

</config>

<description><![CDATA[Lemmatize selected attributes

The lemmatization is configured in the 'configfile' for selected languages. Documents whose language(s) - specified in the 'languages' attribute - are in the set of configured languages will be subject to lemmatization. When there is more than one language for a document, the according language codes must be semicolon-separated.

The 'attributes' parameter specifies the fields to which lemmas are added. Field names are separated by blank. If a field name is in the form source:target, the lemmas are not added to the source field, but written into the field specified as target field.

The 'configfile' parameter defines the name of the configuration file for the mappings of languages on dictionary resources; see the comments in that file for further details.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOSubjectNameNormalizerNew**" type="general" hidden="0">

<load module="processors.SubjectNameNormalizer" class="SubjectNameNormalizer"/>

<config>

<param name="Attribute" value="subjectbasename" type="str"/>

<param name="MatchedField" value="matchedbasename" type="str"/>

<param name="Separator" value="|" type="str"/>

</config>

<description><![CDATA[SubjectNameNormalizer

Normalizes the &amp;lt;Separator&amp;gt;-separated entities in &amp;lt;Attribute&amp;gt;. Mostly it is used

for cleansing navigators, e.g. the companies navigator.

A cleansed companies navigator should only contain &amp;quot;Procter &amp;amp; Gamble Corp.;

Microsoft&amp;quot;. This will be done along all processed documents.

You can cleanse as many attributes as you want. Join multiple entries

with a space in &amp;lt;Attribute&amp;gt;.

Entities are separated by &amp;lt;Separator&amp;gt;

You can determine stopwords (like &amp;#039;inc&amp;#039;, &amp;#039;AG&amp;#039;, &amp;#039;Corp&amp;#039;, ... in companies) in

$FASTSEARCH/resources/entitynormalizer/ENstopwords. Stopwords are case insensitive. If

you start the line with &amp;#039;@&amp;#039; then you can use Python-style regular expressions.

Additionally you can map keywords to phrases in the form &amp;quot;&amp;lt;keyword,...&amp;gt; &amp;lt;phrase&amp;gt;&amp;quot;

like &amp;#039;fast,search,transfer FAST Search &amp;amp; Transfer&amp;#039;

which means, that each entity that contains &amp;#039;fast&amp;#039; and &amp;#039;search&amp;#039; and &amp;#039;transfer&amp;#039; in this order will

end up to &amp;#039;FAST Search &amp;amp; Transfer&amp;#039;. The mappings are stored in

$FASTSEARCH/resources/entitynormalizer/ENmappings. Keywords are case insensitive. If

you start the line with &amp;#039;@&amp;#039; then you can use Python-style regular expressions.

Entities which don&amp;#039;t match to that rules will be catched in the way that the first occurrence will

be the one which is taken for all other matches. Two entity match to each other if their tokens match

(solely word characters, lowercased). Example: &amp;#039;Colgate-Palmolive&amp;#039; matches to &amp;#039;Colgate Palmolive&amp;#039;.

If &amp;#039;Corp&amp;#039; is defined as a stop word then it matches to &amp;#039;Colgate Palmolive Corp.&amp;#039; as well.

All entities are stored by the Cache Manager. If you want to clear the entity table and start from

scratch, just remove the values at the command line with &amp;quot;$FASTSEARCH/bin/cmctrl remove ENentities&amp;quot;.

The MatchedField field shall get the field name that would store the base name that has been mapped

to by one or more documents.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOAssociatedNameNormalizerNew**" type="general" hidden="0">

<load module="processors.SubjectNameNormalizer" class="SubjectNameNormalizer"/>

<config>

<param name="Attribute" value="associatedcompanybasenamenav" type="str"/>

<param name="MatchedField" value="matchedassociatedbasename" type="str"/>

<param name="Separator" value="|" type="str"/>

</config>

<description><![CDATA[SubjectNameNormalizer

Normalizes the &amp;amp;lt;Separator&amp;amp;gt;-separated entities in &amp;amp;lt;Attribute&amp;amp;gt;. Mostly it is used

for cleansing navigators, e.g. the companies navigator.

A cleansed companies navigator should only contain &amp;amp;quot;Procter &amp;amp;amp; Gamble Corp.;

Microsoft&amp;amp;quot;. This will be done along all processed documents.

You can cleanse as many attributes as you want. Join multiple entries

with a space in &amp;amp;lt;Attribute&amp;amp;gt;.

Entities are separated by &amp;amp;lt;Separator&amp;amp;gt;

You can determine stopwords (like &amp;amp;#039;inc&amp;amp;#039;, &amp;amp;#039;AG&amp;amp;#039;, &amp;amp;#039;Corp&amp;amp;#039;, ... in companies) in

$FASTSEARCH/resources/entitynormalizer/ENstopwords. Stopwords are case insensitive. If

you start the line with &amp;amp;#039;@&amp;amp;#039; then you can use Python-style regular expressions.

Additionally you can map keywords to phrases in the form &amp;amp;quot;&amp;amp;lt;keyword,...&amp;amp;gt; &amp;amp;lt;phrase&amp;amp;gt;&amp;amp;quot;

like &amp;amp;#039;fast,search,transfer FAST Search &amp;amp;amp; Transfer&amp;amp;#039;

which means, that each entity that contains &amp;amp;#039;fast&amp;amp;#039; and &amp;amp;#039;search&amp;amp;#039; and &amp;amp;#039;transfer&amp;amp;#039; in this order will

end up to &amp;amp;#039;FAST Search &amp;amp;amp; Transfer&amp;amp;#039;. The mappings are stored in

$FASTSEARCH/resources/entitynormalizer/ENmappings. Keywords are case insensitive. If

you start the line with &amp;amp;#039;@&amp;amp;#039; then you can use Python-style regular expressions.

Entities which don&amp;amp;#039;t match to that rules will be catched in the way that the first occurrence will

be the one which is taken for all other matches. Two entity match to each other if their tokens match

(solely word characters, lowercased). Example: &amp;amp;#039;Colgate-Palmolive&amp;amp;#039; matches to &amp;amp;#039;Colgate Palmolive&amp;amp;#039;.

If &amp;amp;#039;Corp&amp;amp;#039; is defined as a stop word then it matches to &amp;amp;#039;Colgate Palmolive Corp.&amp;amp;#039; as well.

All entities are stored by the Cache Manager. If you want to clear the entity table and start from

scratch, just remove the values at the command line with &amp;amp;quot;$FASTSEARCH/bin/cmctrl remove ENentities&amp;amp;quot;.

The MatchedField field shall get the field name that would store the base name that has been mapped

to by one or more documents.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOEntityNormalizer**" type="general" hidden="0">

<load module="processors.EntityNormalizer" class="EntityNormalizer"/>

<config>

<param name="Attribute" value="companies subjectname" type="str"/>

<param name="Separator" value="|" type="str"/>

</config>

<description><![CDATA[EntityNormalizer

Normalizes the &amp;amp;lt;Separator&amp;amp;gt;-separated entities in &amp;amp;lt;Attribute&amp;amp;gt;. Mostly it is used

for cleansing navigators, e.g. the companies navigator.

A cleansed companies navigator should only contain &amp;amp;quot;Procter &amp;amp;amp; Gamble Corp.;

Microsoft&amp;amp;quot;. This will be done along all processed documents.

You can cleanse as many attributes as you want. Join multiple entries

with a space in &amp;amp;lt;Attribute&amp;amp;gt;.

Entities are separated by &amp;amp;lt;Separator&amp;amp;gt;

You can determine stopwords (like &amp;amp;#039;inc&amp;amp;#039;, &amp;amp;#039;AG&amp;amp;#039;, &amp;amp;#039;Corp&amp;amp;#039;, ... in companies) in

$FASTSEARCH/resources/entitynormalizer/ENstopwords. Stopwords are case insensitive. If

you start the line with &amp;amp;#039;@&amp;amp;#039; then you can use Python-style regular expressions.

Additionally you can map keywords to phrases in the form &amp;amp;quot;&amp;amp;lt;keyword,...&amp;amp;gt; &amp;amp;lt;phrase&amp;amp;gt;&amp;amp;quot;

like &amp;amp;#039;fast,search,transfer FAST Search &amp;amp;amp; Transfer&amp;amp;#039;

which means, that each entity that contains &amp;amp;#039;fast&amp;amp;#039; and &amp;amp;#039;search&amp;amp;#039; and &amp;amp;#039;transfer&amp;amp;#039; in this order will

end up to &amp;amp;#039;FAST Search &amp;amp;amp; Transfer&amp;amp;#039;. The mappings are stored in

$FASTSEARCH/resources/entitynormalizer/ENmappings. Keywords are case insensitive. If

you start the line with &amp;amp;#039;@&amp;amp;#039; then you can use Python-style regular expressions.

Entities which don&amp;amp;#039;t match to that rules will be catched in the way that the first occurrence will

be the one which is taken for all other matches. Two entity match to each other if their tokens match

(solely word characters, lowercased). Example: &amp;amp;#039;Colgate-Palmolive&amp;amp;#039; matches to &amp;amp;#039;Colgate Palmolive&amp;amp;#039;.

If &amp;amp;#039;Corp&amp;amp;#039; is defined as a stop word then it matches to &amp;amp;#039;Colgate Palmolive Corp.&amp;amp;#039; as well.

All entities are stored by the Cache Manager. If you want to clear the entity table and start from

scratch, just remove the values at the command line with &amp;amp;quot;$FASTSEARCH/bin/cmctrl remove ENentities&amp;amp;quot;.]]></description>

<inputs>

</inputs>

</processor>

<processor name="LMCODuplicatesDeleter" type="general" hidden="0">

<load module="processors.DuplicatesDeleterMultivalue" class="DuplicatesDeleterMultivalue"/>

<config>

<param name="Input" value="subjectname subjectcity subjectstate subjectcountry prodservicedesc statute topic violation institutiontype consumeragerange subjectphonenav consumerstate subjecturl subjectemaildomain subjectbasename" type="str"/>

<param name="Output" value="subjectname subjectcity subjectstate subjectcountry prodservicedescnav statutenav topicnav violationnav institutiontypenav consumeragerangenav subjectphonenav consumerstatenav subjecturl subjectemaildomain subjectbasename" type="str"/>

<param name="Separator" value="|" type="str"/>

</config>

<description><![CDATA[Delete duplicates values from a multivalue field

Also removes additional separators for empty values]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOSubjectNameStripper**" type="general" hidden="0">

<load module="processors.RegexChanger" class="RegexChanger"/>

<config>

<param name="changeTo" value="" type="str"/>

<param name="inputField" value="subjectbasename" type="str"/>

<param name="outputField" value="subjectbasenamestripped" type="str"/>

<param name="regex" value="(\s+|;|,|\.)" type="str"/>

</config>

<description><![CDATA[Replace all occurrences of the regex from the input field to &amp;amp;amp;quot;changeTo&amp;amp;amp;quot; and put resulting text into output field (usually both are &amp;amp;amp;amp;apos;data&amp;amp;amp;amp;apos;). Default settings will redact all social security numbers from data (replacing them with myssn). This eliminates dropdowns that can mess up results. Put it after SimpleFormatter. joe meree]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOCopyToSubjectBaseNameStrippedPhonetic**" type="general" hidden="0">

<load module="processors.Basic" class="AttributeCopy"/>

<config>

<param name="Attributes" value="" type="str"/>

<param name="Input" value="subjectbasenamestripped" type="str"/>

<param name="Output" value="subjectbasenamestrippedphonetic" type="str"/>

</config>

<description><![CDATA[Copy document attributes

The document attribute named by the Input configuration parameter

is copied to the attribute named by the Output parameter. Any existing

values in the output is lost. If the input attribute does not exist,

the operation is ignored

Several attributes can be copied using the Attributes parameter. It

has the form &quot;a:b c:d&quot;, where a is copied to b and c is copied to d. Any

number of such mappings can be declared. The Attributes copy is performed

first, then the Input to Output copy.

From attributes starting with a &#039;\*&#039; are looked up in the routing attributes (excluding the leading &#039;\*&#039;), allowing routing attributes to be copied to normal document attributes.]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOPhoneticNormalizer**" type="general" hidden="0">

<load module="processors.linguistics.PhoneticNormalizer" class="PhoneticNormalizer"/>

<config>

<param name="configuration" value="default:etc/phonetic/phonetic\_aggressive.xml:subjectbasenamestrippedphonetic:phonetic4 default:etc/phonetic/phonetic\_medium.xml:subjectbasenamestrippedphonetic:phonetic3 default:etc/phonetic/phonetic\_mild.xml:subjectbasenamestrippedphonetic:phonetic2 default:etc/phonetic/phonetic\_exact.xml:subjectbasenamestrippedphonetic:phonetic1" type="str"/>

</config>

<description><![CDATA[Soundex-like phonetic normalizer

Transforms the content of the fields it is applied to into a

phonetically normalized representation, according to the rules

defined in its configuration file. The idea is to map content to a

canonical form which only preserves the most important phonetic

information (such as the consonants and core vowels). The

configuration can be adjusted to the needs of the language it is

intended to be used on.

A classical example of phonetic normalization is Soundex, others

include Koelner Phonetik and Wiener Phonetik (both variants for

the German language).

Enter the paths to the configuration files into the

&amp;amp;amp;amp;amp;amp;amp;quot;configuration&amp;amp;amp;amp;amp;amp;amp;quot; field below (path specifications should

be relative to your FASTSEARCH root directory). The paths are

entered as space-separated tuples, where each tuple consists of

four things: language, path to the configuration file, source

document field, target document field (these elements are

separated by colon).]]></description>

<inputs>

</inputs>

</processor>

<processor name="**LMCOPhoneticNormalizerSubjectCity**" type="general" hidden="0">

<load module="processors.linguistics.PhoneticNormalizer" class="PhoneticNormalizer"/>

<config>

<param name="configuration" value="default:etc/phonetic/phonetic\_aggressive.xml:subjectcity:phonetic8 default:etc/phonetic/phonetic\_medium.xml:subjectcity:phonetic7 default:etc/phonetic/phonetic\_mild.xml:subjectcity:phonetic6 default:etc/phonetic/phonetic\_exact.xml:subjectcity:phonetic5" type="str"/>

</config>

<description><![CDATA[Soundex-like phonetic normalizer

Transforms the content of the fields it is applied to into a

phonetically normalized representation, according to the rules

defined in its configuration file. The idea is to map content to a

canonical form which only preserves the most important phonetic

information (such as the consonants and core vowels). The

configuration can be adjusted to the needs of the language it is

intended to be used on.

A classical example of phonetic normalization is Soundex, others

include Koelner Phonetik and Wiener Phonetik (both variants for

the German language).

Enter the paths to the configuration files into the

&amp;amp;amp;amp;amp;amp;amp;amp;amp;quot;configuration&amp;amp;amp;amp;amp;amp;amp;amp;amp;quot; field below (path specifications should

be relative to your FASTSEARCH root directory). The paths are

entered as space-separated tuples, where each tuple consists of

four things: language, path to the configuration file, source

document field, target document field (these elements are

separated by colon).]]></description>

<inputs>

</inputs>

</processor>

<processor name="CreateNgrams6" type="general" hidden="0">

<load module="processors.CreateNgrams" class="CreateNgrams"/>

<config>

<param name="Input" value="subjectaddress" type="str"/>

<param name="N" value="6" type="int"/>

<param name="Output" value="addressngramfield6" type="str"/>

<param name="Separator" value="; " type="str"/>

</config>

<description><![CDATA[Create Ngrams using some field contents.

Extracts given field contents and creates Ngrams, then stores them in a output field.]]></description>

<inputs>

</inputs>

</processor>

=======================================================================================================